IOWA STATE UNIVERSITY
Digital Repository

Graduate Theses and Dissertations

Iowa State University Capstones, Theses and Dissertations

2017

# Engineering analytics through explainable deep learning

Sambuddha Ghosal
*Iowa State University*

Follow this and additional works at: https://lib.dr.iastate.edu/etd

 Part of the Agriculture Commons, Computer Sciences Commons, Mechanical Engineering Commons, and the Plant Sciences Commons

Recommended Citation
Ghosal, Sambuddha, "Engineering analytics through explainable deep learning" (2017). *Graduate Theses and Dissertations*. 16277.
https://lib.dr.iastate.edu/etd/16277

This Thesis is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.

**Engineering analytics through explainable deep learning**

by

**Sambuddha Ghosal**

A thesis submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

Major: Mechanical Engineering

Program of Study Committee:
Soumik Sarkar, Major Professor
Baskar Ganapathysubramanian
Arti Singh

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this thesis. The Graduate College will ensure this thesis is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University
Ames, Iowa
2017
Copyright © Sambuddha Ghosal, 2017. All rights reserved.

## DEDICATION

***....Dedicated to my parents***

# TABLE OF CONTENTS

**Page**

# LIST OF FIGURES

**Page**

# ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research throughout my graduate education and the writing of this thesis.

First and foremost, Dr. Soumik Sarkar for his guidance, patience and support throughout this research and the writing of this thesis. His insights and words of encouragement have often inspired me and renewed my hopes for completing my graduate education.

I would also like to thank my committee members, Dr. Baskar Ganapathysubramanian and Dr. Arti Singh for their valuable efforts and contributions in helping me to bring this work to fruition. I would additionally like to thank Dr. James Michael for his guidance throughout the later stages of preparing my thesis.

# ABSTRACT

Pattern recognition has its origins in engineering while machine learning developed from computer science. Today, artificial intelligence (AI) is a booming field with many practical applications and active research topics that deals with both pattern recognition and machine learning. We now use softwares and applications to automate routine labor, understand speech (using Natural Language Processing) or images (extracting hierarchical features and patterns for object detection and pattern recognition), make diagnoses in medicine, even intricate surgical procedures and support basic scientific research.

This thesis deals with exploring the application of a specific branch of AI, or a specific tool, Deep Learning (DL) to real world engineering problems which otherwise had been difficult to solve using existing methods till date. Here we focus on different Deep Learning based methods to deal with two such engineering problems. We also explore the inner workings of such models through an explanation stage for each of the applied DL based strategies that gives us a sense of how such typical black box models work, or as we call it, an explanation stage for the DL model.

This explanation framework is an important step as previously, Deep Learning based models were thought to be frameworks which produce good results (classification, object detection, object recognition to name a few), but with no explanations or immediately visible causes as to why it achieves the results it does. This made Deep Learning based models hard to trust amongst the scientific community. In this thesis, we aim to achieve just that by deploying two such explanation frameworks, one for a 2D image study case and another for a 3D image voxel study case, which will be discussed later in the subsequent chapters.

# CHAPTER 1. INTRODUCTION

In this chapter, a short introduction to the concept of Deep Learning is put forth. Section 1.1 discusses the motivation for carrying out this research and a brief literature survey is presented in section 1.2.

## 1.1 Motivation

Deep Learning (referred to as "DL" from here on), in general is a very powerful tool that deals with solving tasks that are easy for people to perform but at the same time hard for people to describe formally i.e., problems that are more intuitive, for instance, recognizing words or faces in in images. This kind of methods allow computers to learn from experience and understand their surroundings in terms of hierarchy of concepts, with each concept defined through its relation to simpler concepts. This hierarchy of concepts enables the computer to learn complicated concepts by building them out of simpler ones.

In recent times, DL models have been shown to outperform other state of the art techniques in handling and analyzing large dimensional data (both spatial and temporal), by learning the hierarchical features to perform various tasks such as, classification and bulk structure detection given a large corpus of 2-dimensional (2D) data or images. As an extension, embedding of 3-dimensional (3D) spatiotemporal data (where the data have spatial features evolving over time) has also been performed using a 3D Convolutional Neural Network (3D CNN) framework.

## 1.2 Literature Survey

Deep Learning has proved its mettle in various fields of modern science such as medical imaging applications that have achieved dermatologist level classification accuracies for skin cancer (1), in modeling neural responses and population in visual cortical areas of the brain (2) and in predict-

ing sequence specificities of DNA- and RNA-binding proteins (3). Similarly, deep learning based techniques have made transformative demonstration in the context of performing complex cognitive tasks such as achieving human level or better accuracy for playing Atari games using Deep Q network (4) and beating a human expert in playing the Chinese game of Go (5).

### 1.2.1 Object Recognition via Handcrafting of Features

Appearance of an image region has been effectively described by this method, although it has proved to be time-consuming and resource intensive. SIFT (Scale Invariant Feature Transform)(6), HoG (histogram of Gradients)(7), LBP (Local Binary Pattern)(8) are some low-level techniques that follow handcrafting. These essentially describe the low order statistics of the edge distribution where each image region generate fixed size vector to describe properties of the region. SIFT describes the appearance of an image, HoG describes edges and LBP describes textures. Thus, depending on the task at hand, these techniques can be used separately (9)(10)(11) or can be combined (15).

Probabilistic graphical models (12, 13, 14) have also been successful in various image-recognition tasks by building on low level handcrafted features. Probabilistic models can model object shapes, appearance, occlusion and relative scales quite efficiently. One such modeling technique is to learn the parameters of a model to maximize object likelihood in an image with expectation-maximization (EM) algorithm. A "generic" knowledge learnt by graphical models can even be applied to models of unrelated categories as "prior" knowledge.

### 1.2.2 Object Detection via Handcrafting of Features

Object detection can be performed using handcrafted features. Detection systems have been built based on "part-based" models (10, 15, 16, 17, 18, 19, 20) with patch appearance features like HoG (Histogram of Gradient) (7), SIFT (Scale Invariant Feature Transform) (6), LBP (Local Binary Pattern) (8). These are general object detection models, an significant achievement in

computer vision that requires prediction of object labels and object locations at the same time. Pedestrian detection with HoG features (7) is one such example.

Different object detection results can enhance or inhibit each other within a scene or context. Active research in object recognition has been carried out on this part (21, 22, 23, 24, 25, 26, 27, 28). (21) discusses on end-to-end training of a multi-class detector and post-processing

### 1.2.3 Neural Networks applied in Object Detection and Feature Learning

Neural networks (regarded as deep architectures) learn hierarchies of features with increasing levels of invariance and complexity. They are suited for vision tasks that demands making sense of intricately complex features (29)(30). In multi-layer networks, trained end-to-end with limited prior knowledge, feature representations are jointly learnt with classifiers. Convolutional Neural Networks (CNNs) (31) can be applied as a multi-layer neural network with end-to-end supervised learning capabilities that can learn from raw image pixels.

Unsupervised learning techniques have also had successes when applied for training or pre-training multi-layer networks. Some unsupervised learning techniques are Stacked Auto-Encoders (32) and Restricted Boltzmann Machines (33). The idea of auto-encoders was proposed in (32), which initialize network layers directly by minimizing reconstruction error of input from output. After initialization of each layer, the entire network is jointly fine-tuned with back-propagation based on the label information. This layer-by-layer network initialization was introduced by (33).

For image classification purposes convolutional networks (31) has been very successful over the past few years. A few cases where it has been applied are: handwritten characters (31), house numbers (34)(35)(36), images of objects (36)(37), traffic signs (34)(35) among several other applications. CNNs have had instrumental success to carry out a 1000-class classification on the ImageNet dataset (39). Figure 1.1 illustrates the CNN devised for carrying out the ImageNet classification.

(32) combines structure learning with a CNN for classifying individual digits and then train with hand written strings of digits. A problem with networks with a large number of parameters is

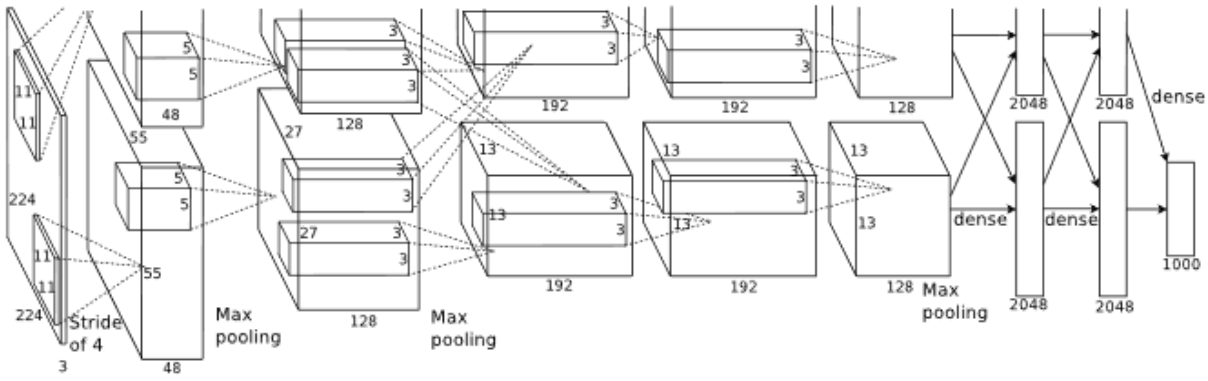



Figure 1.1 Reproduced from (39): The CNN architecture crafted for the 1000 category ImageNet classification, explicitly showing the delineation of responsibilities between two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The networks input is 150,528-dimensional, and the number of neurons in the networks remaining layers is given by 253, 440 - 186, 624 - 64, 896 - 64, 896 - 43, 264 - 4096 - 4096 - 1000.

that it can easily over-fit to training data. To overcome this problem, several techniques have been developed. For instance, there are data augmentation algorithms (39)(35) and network regularization algorithms (39)(40)(41)(36). Data augmentation and regularization are usually combined to achieve desirable results when training neural networks.

To circumnavigate over-fitting issues, $l_2$ regularization can be a simple and effective approach, there are other forms of network regularization as well, namely bayesian methods (40), weight elimination (41) and early stopping. Dropout (42) is another recent regularization method introduced by Hinton et al. This involves randomly deleting a certain proportion (specified by a hyper-parameter) of the activations in each layer. Experiments have shown that Dropout significantly reduces over-fitting issues and improves testing performance.

## 1.3 Thesis Organization

In Chapter 3, we develop a 2D Deep CNN (Convolutional Neural Network) model that deals with detection and classification of Soybean Leaf diseases from a dataset of leaf images. The model architecture is formulated is trained on several images, then validated and finally tested on a previously unseen batch of images on which the performance of the trained model was found to be significantly good. We then apply a visual explanation framework to get a sense of how the model works by applying an explanation tool, namely, Grad-CAM (Gradient based Class Activation Mapping) to determine how the model behaves when it actually does the prediction. This leads us to explain the behavior of black box models such as a Deep CNN.

In Chapter 4, we further develop our framework to deal with not only spatial data, but further extend it to account for temporal variations of such spatial data, thus, effectively dealing with highly complex spatio-temporal data, which can be difficult to deal with the existing tools. Thus we come up with a novel 3D Deep Convolutional Neural Network (DCNN) architecture and use it to detect early and predict the onset on combustion instabilities in enclosed combustion environments and this is done by training our model on a set of sequential combustion flame image snapshots captured for different Air-Fuel ratio protocols. We validate and test our model and it perfoms quite efficiently in early detection of the combustion instabilities. As an explanation framework for this 3D black-box model, we use a different approach than our problem in Chapter 3. For this particular study, we use 3D Convolutional Auto-Encoder (CAE) to actually visualize the coherent structures that prove that the predicted zones are truly zones of flame instabilities.

Finally, in Chapter 5, we put forward concluding remarks to our work and also discuss some open questions as well as further works based on what has already been done and presented in this thesis, which is currently in progress.

# CHAPTER 2. PRELIMINARIES ON DEEP LEARNING AND TRAINING CONSIDERATIONS

This Chapter provides a general discussion on the different DL techniques devised to carry out our target objectives i.e, to perform the prediction, detection and classification tasks presented in CHAPTERS 3 and 4. We also discuss in detail, the explanation techniques mentioned earlier in the abstract (namely, Grad-CAM and CAE) applied to make sense of the DL (Deep CNN) black-box model for both 2D and 3D cases. Additionally, some training considerations taken into account while building the developed models have been discussed in this chapter.

This chapter is organized as follows: In section 2.1, we talk about Restricted Boltzmann machines, which are the fundamental units of Deep Neural Networks. In section 2.2, a general discussion of Deep Neural Networks is given, followed by a discussion of Convolutional Neural Networks in section 2.3. Then we discuss briefly on the explanation frameworks: In section 2.4, explanation framework applied in CHAPTER 3 is discussed and in section 2.5, the explanation framework applied in CHAPTER 4 is discussed.

## 2.1 Restricted Boltzmann Machines (RBMs)

A Restricted Boltzmann machine (RBM) is a generative stochastic artificial neural network that learns features in an unsupervised manner based on a probabilistic model [43]. It became popularized by Geoffrey Hinton in the mid-2000s (2006), when he and his group developed fast learning algorithms to effectively train RBMs [43]. RBMs have been used in a diverse area of applications as a means to reduce data dimension [44], collaborative filtering [45], solving classification problems [46, 47], topic modeling [48, 49, 50] and feature learning [51]. The success of RBMs largely owes to the fact that the extracted features is nonlinear in nature. As a result, it often generates good results when used in conjunction with a linear classifier such as a support vector machines

(SVM) or a perceptron. An RBM basically attempts to maximize the likelihood of the data using a particular graphical model and typically employs the learning algorithm via stochastic maximum likelihood. Via this method, it is capable of capturing persistent regularities from the data and learning a probability distribution over the set of the provided inputs. However, a caveat is that, to effectively train an RBM model, we require a sufficiently large dataset.

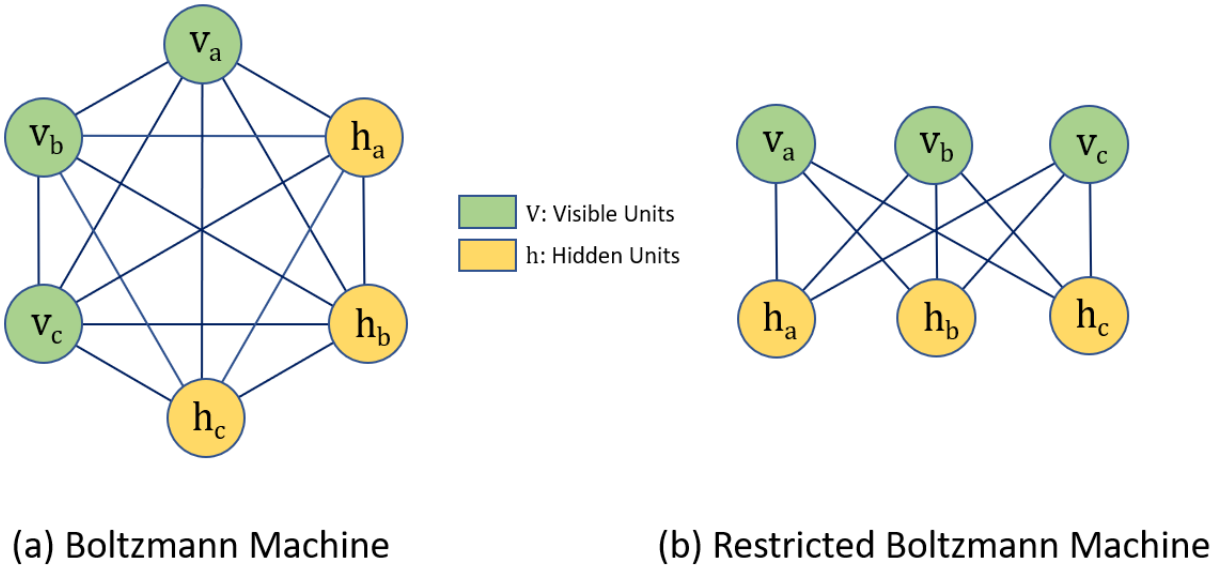



Figure 2.1 Differentiation between Boltzmann machines and Restricted Boltzmann machines (RBMs). In RBMs, there are no interconnections between the nodes in the same layer i.e., there are no *visible-visible* or *hidden-hidden* connections.

Restricted Boltzmann machines are actually a variant of Boltzmann machines [52]. In a Boltzmann machine, nodes, or neurons in the same group are connected in addition to nodes from the other group (see Figure 2.1). Typically, two groups are present in a Boltzmann machines: the visible and hidden units. However, the interconnectedness of all the neurons within and between groups may complicate modeling. Hence, a restricted version of the Boltzmann machine is used with a constraint that the neurons must form a bipartite graph where there are no connections within a group. This restriction make learning easier, as the hidden units become conditionally in-

dependent given the visible states [53]. RBMs are also the building blocks to deep neural networks (DNN), where they can be stacked to increase the modeling capacity.

## 2.2 Deep Neural Networks

Stacking multiple layers of RBMs together results in the class of architectures known as Deep Neural Networks. Figure 2.2 illustrates this. Increasing the number of hidden layers in a network increases its nonlinear modeling capacity. Deep neural networks are also known as artificial neural networks (ANN), which are inspired by the observations and biological models proposed by Harvard neurophysiologists David H. Hubel and Torsten Wiesel. From these observations, they showed how the visual system of living animals builds complex representations from simple stimuli which established the fundamental concepts of deep learning [54, 55, 56, 57].

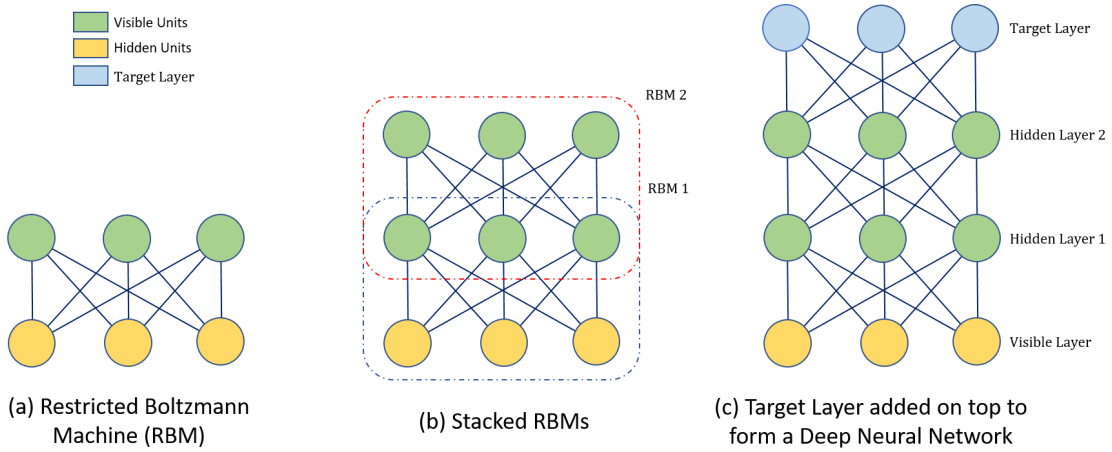



Figure 2.2 Stacking of RBMs to form a Deep Neural Network

In deep neural networks, each layer of the neurons trains on different sets of features using the outputs from the previous layer. The deeper we advance into the network, the higher the complexity of the features that the network can recognize. For example, in the application of face recognition, the first layer of the network usually learns primitives such as simple edges and curves. As we move on to the intermediate layers, the hidden layers begin learning a combination of these primitives, such as the eyes, the mouth, the ears, and the nose. The deepest layers are capable of

combining the parts and begin recognizing faces (see Fig. 2.2). This concept widely known as the hierarchy of features, that is, a hierarchy of features with increasing complexity and abstraction. Therefore, deep networks are suited for handling a large of amount of very high-dimensional data sets.

## 2.3 Deep Convolutional Neural Networks (DCNNs)

DCNNs are suitable machine learning models because of their widely demonstrated efficacy in performing large-scale image classification, automated feature learning capability and ease of training (39). Although the details of DCNN architecture and training procedures have been well described in recent literature (39) (58), we provide a brief description here for the sake of completeness. Compared with fully connected (FC) deep neural networks with the same number of hidden layers, DCNNs achieve a similar level of performance with fewer parameters to learn (39) (59). DCNNs are designed to exploit the two-dimensional (2D) structure of an input image by preserving the locality of features via the utilization of spatially local correlations of an image by using tied weights, which are invariant to the translation of the feature positions(39) (60). Weight sharing among different locations in an image also increases the efficiency of learning because the number of learnable parameters during training are substantially fewer than those in an FC neural network.

In DCNNs, data are represented by multiple feature maps in each hidden layer. Feature maps are obtained by convolving the input image by using multiple filters in the corresponding hidden layer. In other words, they are obtained by repeatedly applying a function across sub-regions over the entire image, i.e., a convolution operation of the input image with a filter. To further decrease the dimension of the data, these feature maps typically undergo non-linear down-sampling with a 2X2 max-pooling operation (61). Max-pooling partitions (or super-pixelates) the input image into sets of non-overlapping rectangles and uses the maximum value for each partition as the output. Because neighboring pixels in an image share similar features, these pixels can be discarded to overcome memory constraints and decrease training time. Furthermore, both spatial and feature

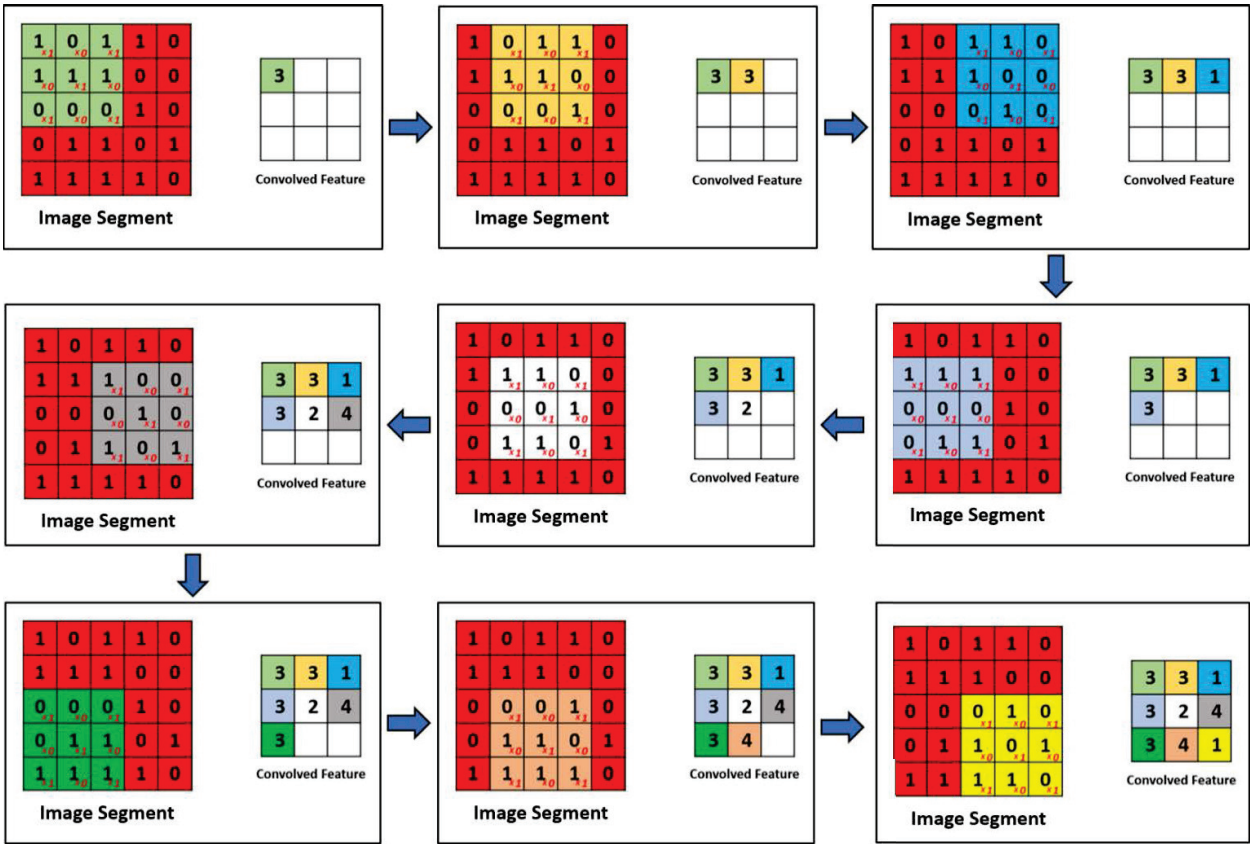



Figure 2.3 Illustration showing how 2D Convolution works

abstractness can be increased by max-pooling, which results in increased position invariance for the filters (61, 62).

To improve the performance of the architecture, a Batch Normalization layer is added between the two neuron layers, which normalizes the activations of the previous layer at each batch, i.e., applies a transformation that maintains the mean activation close to 0 and the activation standard deviation close to 1 (63).

After max-pooling, multiple dimension-reduced vector representations of the input are acquired, and the process is repeated in the next layer to give a higher-level representation of the data. At the final pooling layer, the resultant outputs are linked to the FC layer, where Rectified Linear Unit ($ReLU$) activation outputs (64) from the hidden units are joined to output units to infer a

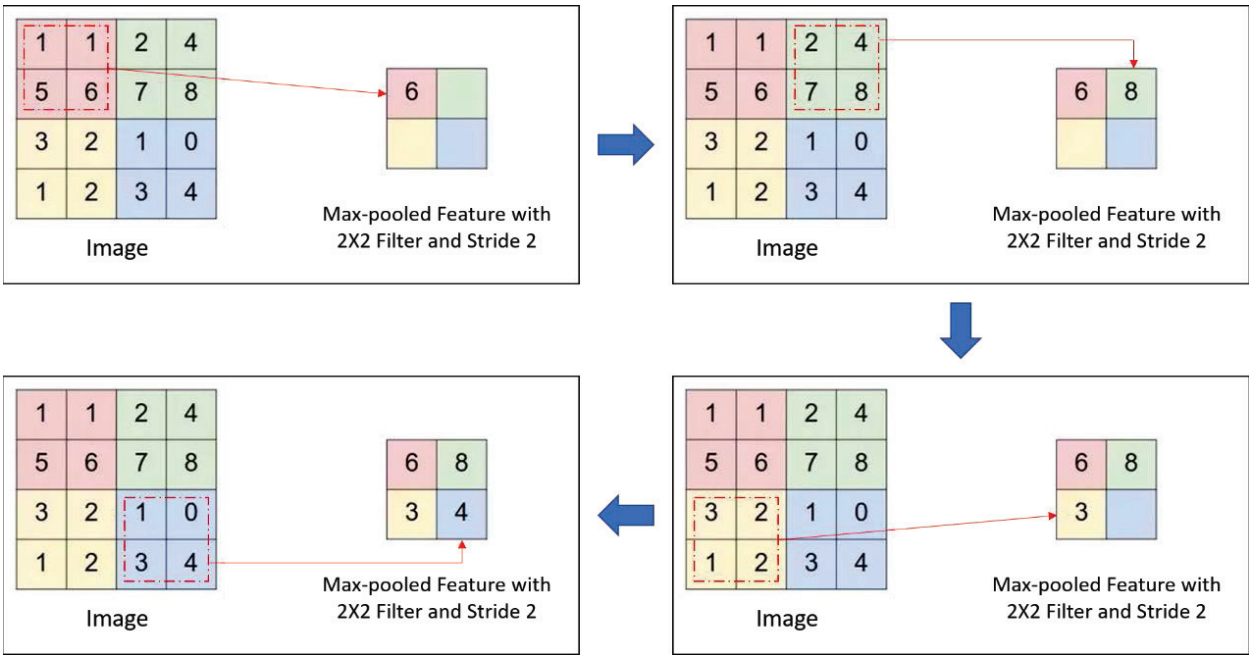



Figure 2.4 Illustration showing how 2D Max-Pooling works

predicted class on the basis of the highest joint probability given the input data. With this setup, the probability of an input vector $v$ being a member of the class $i$ can be written as follows:

$$Pr(Y = i|\mathbf{v},\ \mathbf{W},\ \mathbf{b}) = softmax_i(\mathbf{W}\mathbf{v}\ +\ \mathbf{b}) = \frac{e^{W_i v + b_i}}{\sum_j e^{W_j v + b_j}} \tag{2.1}$$

where elements of $\mathbf{W}$ denote the link weights and elements of $\mathbf{b}$ denote the biases. The model prediction is the class with the highest probability:

$$y_{pred} = argmax_i Pr(Y = i|\mathbf{v},\ \mathbf{W},\ \mathbf{b}) \tag{2.2}$$

The model weights, $\mathbf{W}$, and biases, $\mathbf{b}$, are optimized by the well-known error backpropagation algorithm (65), wherein true class labels are compared against the model prediction by using an error metric, which becomes the loss function for the (weights and biases) optimization process. The loss function, chosen to be minimized for the dataset $\mathbf{V}$, is the categorical cross-entropy function (66), $\mathcal{L}$, and is given as follows:

$$\mathcal{L}(V,Y) = -\frac{1}{n}\sum_{i=1}^{n} \mathbf{y}^{(i)} ln\ \mathbf{a}(\mathbf{v}^{(i)}) + (1-\mathbf{y}^{(i)})\ ln\ (1-\mathbf{a}(\mathbf{v}^{(i)})) \tag{2.3}$$

Here, $\mathbf{V} = \{\mathbf{v}^{(1)},,\mathbf{v}^{(n)}\}$ is the set of input examples in the training dataset, and $\mathbf{Y} = \{\mathbf{y}^{(1)},,\mathbf{y}^{(n)}\}$ is the corresponding set of labels for those input examples. The $\mathbf{a}(\mathbf{v})$ represents the output of the neural network given input $\mathbf{v}$.

### 2.3.1 Overview of the different layers used in DCNNs

In this section, a short description of the different layers used in developing and training a Deep Convolutional Neural Network model is presented. An illustration of a typical CAE framework is given in Figure 2.5.

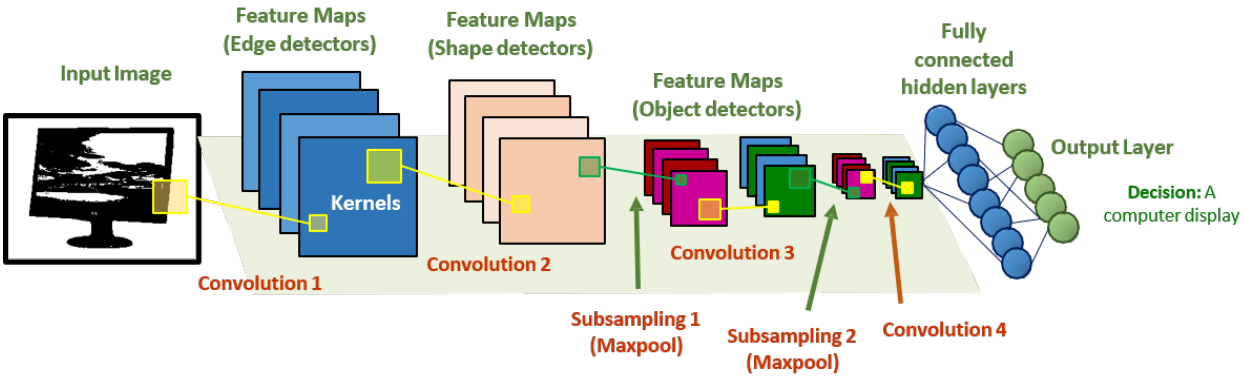



Figure 2.5 Illustration of a typical Convolutional Neural Network architecture

#### 2.3.1.1 The Convolutional layer

The function, "Convolution" is essentially a sliding function (a sort of filter) applied to a 2D or 3D input matrix. The layer associated with performing the convolution function is termed the "Convolutional Layer" and it plays the most important role in building and learning DCNNs. It consists of filters (or kernels) that can be learnt through training. Such filters have small receptive fields to extract local structures and uses shared weights. The filters may specialize in detecting different features. The activations of different filters will depend on the specific type of feature at

some spatial location in the input image. As the filters are convolved over the input, feature maps are generated (as illustrated in Figure 2.3). Stacking the feature maps along the depth dimension gives rise to the full output volume of the convolutional layer.

There are three most prominent hyper-parameters that control the size of the output volume. With a deep layer, many neurons connect to a particular region in the input volume. The neurons will learn to activate different depending on the features in the input. For instance, neurons in the same layer may become active in the presence of edges in different orientations and lie in the same feature hierarchy. Large convolution strides will result in a smaller feature map due to less overlapping receptive fields, whereas small convolution strides (such as 1 X 1 $\rightarrow$ 1 unit along the horizontal and 1 unit along the vertical direction) convolution will result in strongly overlapping receptive fields and, subsequently, larger feature maps. In all cases, the dimensions will be reduced depending on the size of the convolution strides. However, it is sometimes desirable to preserve the original spatial dimensions of the input volume. To achieve this, the borders of the input volume can be padded with zeros (termed as zero-padding) such that after the convolution operation, the reduced spatial dimensions are compensated to maintain original dimensions.

#### 2.3.1.2 The Pooling layer (max-pooling)

Pooling is a form of nonlinear downsampling. Common practices employ the maxpooling scheme (illustrated in Figure 2.4), where a 4 X 4 matrix for example is downsampled into a 2 X 2 one by selecting the element with the highest value in the matrix. Other pooling functions exist too; one may downsample by averaging the values or even computing the $l2$-norm. Pooling is useful because it removes redundancies and helps reducing the dimensions of the data. The intuition is that once a feature is detected, the exact location of the feature may not be as important as the approximate location relative to other features. Doing so also reduces the number of learnable parameters to combat overfitting as well as reducing computation time. Note that using pooling layers is up to the discretion of the user; it is a common practice to periodically insert a pooling layer after several convolutional layers. An additional benefit that pooling offers is the translation invariance

of features. However, most studies are gravitating towards using smaller filters [67] and discarding the pooling layer [68] in order to prevent an excessively aggressive reduction in dimension.

#### 2.3.1.3 The Fully Connected (FC) layer

After several reductions in dimensions from convolution and pooling, the location information of the features become less important. Hence, we can connect the feature maps generated by the filters to the fully connected layers to increase modeling capacity. We can think of the feature maps being vectorized as an input to a single neural net layer. From this point onwards, the forward pass will be similar to the procedure outlined in Section 2.1.2. In the context of classification, a softmax function can be applied on the sigmoid activations of each output neuron to obtain a probability distribution, where the class with the highest probability is selected as the class prediction. Similarly, one can choose to minimize the loss function (such as negative log-likelihood) and optimize the model parameters via gradient descent.

## 2.4 Gradient-weighted Class Activation Mapping (Grad-CAM)

The Grad-CAM method increases the transparency of DCNN-based models and explainability by visualizing the input regions that are more important than others. Based on this, the DCNN then makes predictions. Therefore, Grad-CAM provides explanations for the behavior of our DCNN model as it makes class predictions. Although detailed mathematical formulations and algorithm descriptions can be found in (69), we provide a brief description below for completeness.

Grad-CAM follows the CAM approach to localization (70). This method enables modification of image classification DCNN architectures, in which FC layers are replaced with convolutional layers. Subsequent global-average pooling (71) yields class-specific feature maps. A new method that circumvents issues of CAM combines feature maps that do not require any modification in the network architecture. This method uses a gradient corresponding to a certain class that is fed into the final convolutional layer of a DCNN to produce an approximate localization (heat) map of the important regions in the image for each class. To obtain the Grad-CAM localization

map $L^c_{Grad-CAM}$ for any class $c$, the gradient of the score for class $c$, $y_c$, (before the output layer) with respect to feature maps $A_k$ of a convolutional layer, i.e., $\frac{\partial y^c}{\partial A^k}$, is first computed. The neuron importance weights, $\alpha^c_k$ are then obtained by global-average pooling:

$$\alpha^c_k = \frac{1}{Z}\sum_i \sum_j \frac{\partial y^c}{\partial A^k_{ij}} \tag{2.4}$$

This weight $\alpha^c_k$ represents the partial linearization of the deep network down from A and extracts the importance of feature map $k$ for a target class $c$. The localization map is then obtained by using a weighted combination of forward activation maps and adding a $ReLU$ to obtain the following:

$$L^c_{Grad-CAM} = ReLU(\sum_k \alpha^c_k A^k) \tag{2.5}$$

This result produces a coarse heat-map of the same size as the convolutional feature maps (2X2 for the last convolutional layer of our DCNN architecture). $ReLU$ is applied to the resulting linear combination of maps. $ReLU$ is applied to the resulting linear combination of maps that highlights only the features that positively influence the class of interest, i.e., the pixels which with increasing intensities result in increase of $y^c$. $ReLU$ elicits this result by removing the undesirable negative pixels that may belong to other classes/categories in the image.

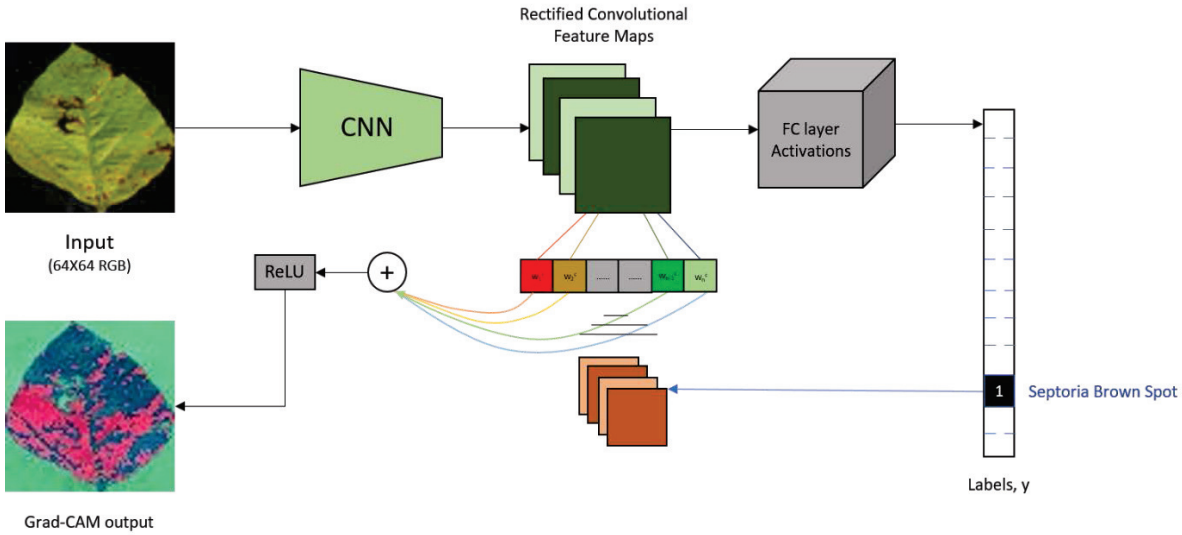



Figure 2.6 Illustration of the Grad-CAM algorithm

We use this unique method in our problem described in CHAPTER 3 to generate heat maps on the images of our test set to determine the extent to which a leaf is diseased. We also use the heat map as a surrogate of disease severity, which we express as a percentage based on the ratio of the diseased leaf area (calculated by counting the number of pixels on the leaf image that are deemed most important, i.e., marked in red, by the Grad-CAM method) to the total leaf area in each image. Figure 2.6 illustrates the workings of this method.

## 2.5 Convolutional Autoencoders

An autoencoder tries to learn the approximation to the identity function in an unsupervised manner, such that the reconstruction of the input is similar to the actual input. In unsupervised learning, only unlabeled data is used. At first glance, it seems to be trivial to learn the identity function. However, this problem becomes not so trivial anymore if we impose some constraints to the learning process such that the algorithm can discover interesting or meaningful underlying patterns in the input data. An illustration of a typical CAE framework is given in Figure 2.7.

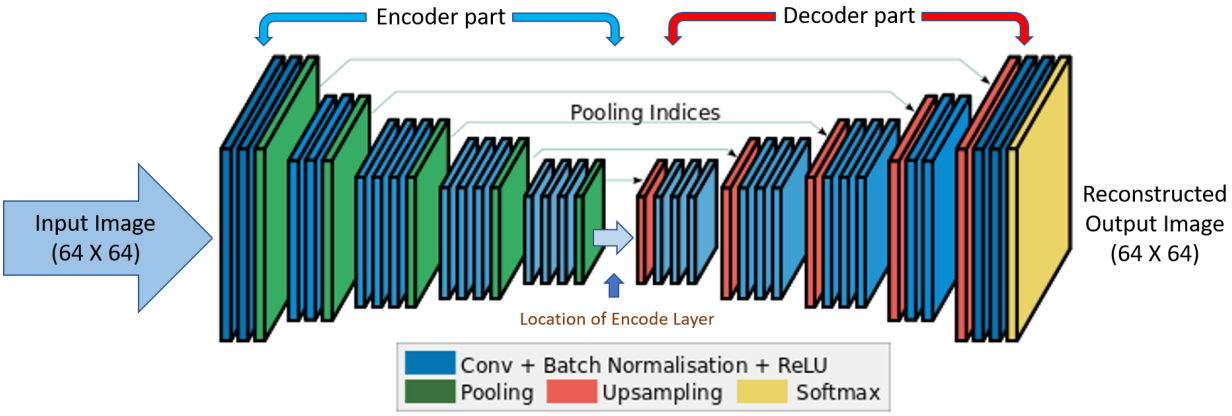



Figure 2.7 Illustration of a typical Convolutional Auto-Encoder architecture

For instance, in case of images, more specifically, in a natural 2D image are often correlated in terms of color. Objects in the image will also have a series of pixels that form the edges of the object. Autoencoders are algorithms that can automatically discover these correlations. Similar to principal component analysis (PCA), autoencoders can learn low-dimensional representation of the

inputs by capturing the codes within the data; in fact, the optimal solution to an autoencoder is strongly related to one found with PCA if linear activations or only a single sigmoid hidden layer are used [72]. The added advantage of autoencoders is that they can be stacked to form stacked autoencoders (SAE), another deep architecture that has a superior nonlinear modeling capacity compared to a single layer of autoencoder or PCA [73].

## 2.6 Training Considerations

### 2.6.1 Data Preprocessing

Data preprocessing plays an essential part before feeding input data into a neural network. Essentially a zero-mean data is desired as an input to a neural network, as it helps in faster model training and validation. For instance, consider what happens when the input to a neuron is always positive. In such a case, the gradients for updating the weights are always all positive or all negative. This calls for the data to be zero-mean. In practice, apart from making the data zero-mean and normalizing, there are other preprocessing techniques like PCA and Whitening. Figure 2.8 illustrates a few data preprocessing schemes.

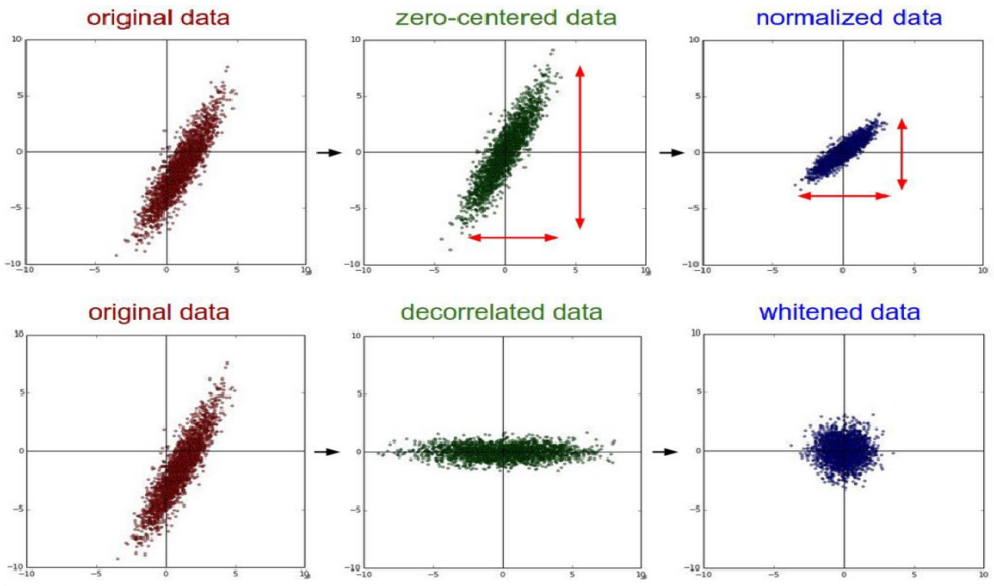



Figure 2.8 Data Preprocessing techniques. *Illustration Courtesy: Stanford CS231n lecture notes*

### 2.6.2 Batch Normalization

Introduced first in (63), Batch Normalization is a form of regularization that in essence lessens the need for other forms of regularization in a neural network. The idea behind batch normalization is to normalize the inputs of each layer in such a way that they have a mean output activation of zero and standard deviation of one (i.e, make unit gaussian activations after each convolution or pooling layer before feeding the output to the next layer). This is analogous to how the inputs to networks are standardized. A known fact is that normalizing the inputs to a network helps it learn. But a network is just a series of layers, where the output of one layer becomes the input to the next. That means we can think of any layer in a neural network as the first layer of a smaller subsequent network. Thought of as a series of neural networks feeding into each other, normalizing the output of one layer before applying the activation function, and then feed it into the following layer (sub-network), helps the learning process for the network, easier, better and faster. Figure 2.10 shows what Batch Normalization does to input data.

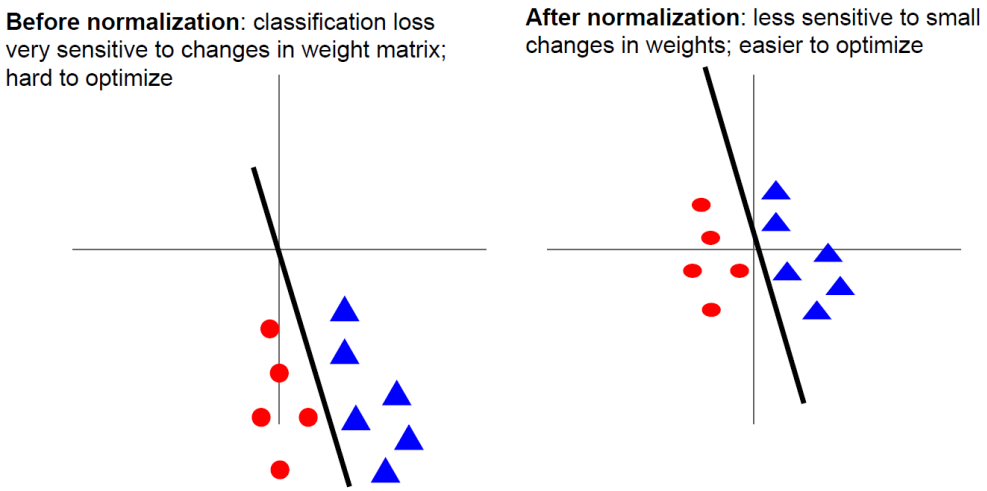



Figure 2.9 The idea behind performing Batch Normalization. *Illustration Courtesy: Stanford CS231n lecture notes*

### 2.6.3 Regularization: Dropout

Dropout is a regularization technique for reducing overfitting in neural networks by preventing complex co-adaptations on training data. The term dropout refers to dropping out units (both hidden and visible) in a neural network. It refers to ignoring activation units (i.e. neurons) during the training phase of certain set of neurons, chosen at random. It was first introduced in (74). More technically, At each training stage, individual nodes are either dropped out of the net with probability $(1 - p)$ or kept with probability $p$, so that a reduced network is left. Incoming and outgoing edges to a dropped-out node are also removed. This is essential in a network with a lot of learning parameters. In a neural network, a fully connected layer may occupy most of the parameters, and hence, neurons develop co-dependency amongst each other during training which curbs the individual power of each neuron leading to over-fitting of training data. Dropout takes care of this issue.

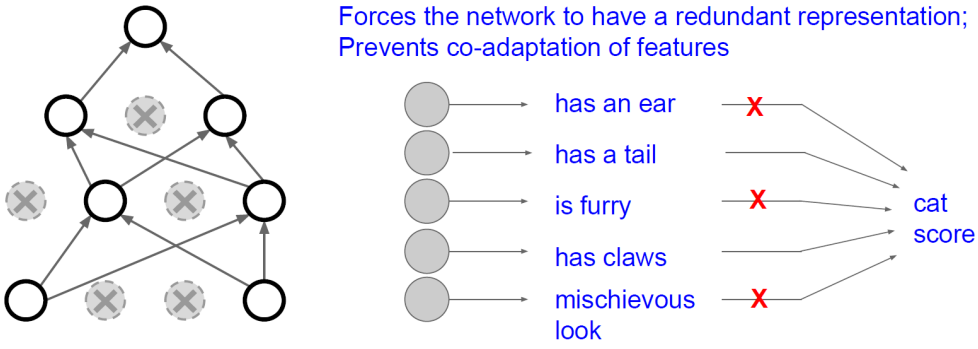



Figure 2.10 Dropout takes care of over-fitting issues by randomly dropping activation units. *Illustration Courtesy: Stanford CS231n lecture notes*

### 2.6.4 Activation Functions

Fundamentally, neural networks are layered collections of nodes, each of which receives a set of inputs and individually makes the decision whether to fire, and propagate information downstream to subsequent layers. The inputs are combined with weights and biases local to each node, which are updated by a learning algorithm in response to the observed error on training examples. This

enables patterns in data to be learned by the neural network, where the value of the weight is proportional to its importance.The weighted input data from all sources is summed to produce a single value, (called the linear combination), which is then fed into an activation function that turns it into an output signal. Conceptually, the activation function is what makes decisions: when given weighted features from some data, it indicates whether or not the features are important enough to contribute to a classification. Hence the purpose of the activation function is to introduce non-linearity into the neural network. This allows the model to generate a response variable that varies non-linearly with its explanatory variables (one where the output couldnt be reproduced from a linear combination of inputs).

For our purposes, we used the Rectified Linear Unit (ReLU) function as the activation function of choice as it has significant advantages over other choices. It not only speeds up training but also for ReLU, the gradient computation is very simple. Computation step of a ReLU is easy as well. In the context of deep neural networks, the rectifier is an activation function, defined as:

$$f(x) = max(0, x) \tag{2.6}$$

where $x$ is the input to a neuraon. A unit utilizing the rectifier is termed a rectified linear unit (ReLU).

# CHAPTER 3. EXPLAINABLE AND APPLIED 2D DEEP LEARNING: BRINGING CONSISTENCY TO PLANT STRESS PHENOTYPING

Plant disease identification based on visual symptoms has predominately remained a manual exercise performed by trained pathologists, primarily due to the occurrence of confounding symptoms. However, the manual rating process is tedious, time-consuming and suffers from inter- and intra- rater variabilities. Our work resolves such issues via introducing the concepts of explainable deep machine learning to automate the process of plant stress identification, classification and quantification. We not only construct a very accurate model that can deliver trained pathologist-level performance, but also explains which visual symptoms it uses to make the prediction. We demonstrate that our method is applicable to a large variety of biotic and abiotic stresses as well as is transferable from one plant to another.

Current approaches for accurate identification, classification and quantification of biotic and abiotic stresses in crop research and production are predominantly visual and require specialized training. However, such techniques are hindered by subjectivity resulting from inter- and intra-rater cognitive variability. This translates to erroneous decisions and a significant waste of resources. Here, we demonstrate a machine learning framework's ability to identify and classify a diverse set of foliar stresses in soybean [Glycine max (L.) Merr.] with remarkable accuracy. We also present an explanation mechanism using gradient-based class activation mapping that isolates the visual symptoms used by the model to make predictions. This unsupervised identification of unique visual symptoms for each stress provides a quantitative measure of stress severity, allowing for identification (type of foliar stress), classification (low, medium or high stress) and quantification (stress severity) in one framework. We reliably identified and classified several biotic (bacterial and fungal diseases) and abiotic (chemical injury and nutrient deficiency) stresses by learning from over 25,000 images. The learnt model appears to be agnostic to species and make good

predictions for other (non-Glycine max) species, demonstrating an ability of transfer learning. The availability of an explainable model that can consistently, rapidly and accurately identify and quantify foliar stresses would have significant implications in scientific research, plant breeding and crop production. The trained model could be deployed in mobile platforms (e.g., unmanned air vehicles and automated ground scouts) for rapid, large-scale scouting or as a mobile application for real-time detection of stress by farmers and researchers.

Conventional plant stress identification and classification has invariably relied on human experts identifying visual symptoms as a means of categorization (75). This process is admittedly subjective and error-prone. Computer vision and machine learning have the capability of resolving this issue and enable accurate, scalable high-throughput phenotyping.

Here, we build a deep learning model that is exceptionally accurate in identifying a large class of soybean stresses from RGB images of soybean leaves (see Figure 1). However, this type of model typically operates as a black-box predictor and requires a leap of faith to believe its predictions. In contrast, visual symptom-based manual identification provides an explanation mechanism (e.g., visible chlorosis and necrosis are symptomatic of iron deficiency chlorosis [IDC]) for disease identification. The lack of explainability is endemic to most black-box models and presents a major bottleneck to their widespread acceptance (76). Here, we sought to look under the hood of the trained model to explain each identification and classification decision made. We conducted this examination by extracting the visual cues or features responsible for a particular decision and determining which region of the leaf image is used by the Deep Convolutional Neural Network (DCNN) model to make a decision and whether this region is correlated with the human-identified symptoms of a particular disease. Our explanation framework is based on concepts of gradient-based class activation mapping (Grad-CAM) (69, 77), which queries each prediction of the trained model to extract visual cues (see Figure 2; also see CHAPTER 2 Section 2.4). Figure 3 illustrates this explanation framework with representative examples from each stress. The trained DCNN correctly identified each stress (top row) on the basis of the input image (second row). The explanation framework then (without any supervision) isolated the visual cues (i.e., most important pixels)

used by the DCNN for stress identification. These regions are highlighted in red in row 3. With statistical significance, the visual cues identified by the explanation framework correlated with the regions exhibiting visual disease symptoms, as assessed by an expert plant pathologist (see Figure 4(b)). The expert rating procedure is discussed as follows:

### 3.1 Severity Rating by Human Experts using a Visual Application

Multiple expert raters were also given a subset of images from our test set to rate the leaf images according to their disease severity. Thus, we obtained severity ratings based on the CNN-Grad-CAM method alongside severity metrics based on the expert raters. We used a custom built app to enable the expert raters to quickly, and efficiently mark severity measures.

Because ease of use was a priority, we chose to build a standard web application so that users could access the app immediately, on any device, without installation. We also chose not to use any particular frameworks for the front-end or back-end, instead using standard JavaScript and PHP. We used a standard MySQL database to store the marked image data. One developer created an initial prototype of the front-end in a day, which proved very useful for developing a shared vision of the final app (see Figure 3.1).

The front-end was plain HTML and JavaScript with some AJAX calls into a PHP backend. We used commonly used, well-supported technologies. We used the Raphael JS library to draw vector graphics on top of the leaf images, with simple JavaScript event handlers to simulate drawing on the image. To support touch-based mobile devices, we used the jQuery Touch Punch library to map touch events to mouse events. On the backend, the marked image data were stored by converting the path data to a JSON string and saving the data in a MySQL BLOB-type column. To track multiple users, we had the app open with an enter your user-name field. The user-name was then included in subsequent back-end requests. Because the app was used only internally and for a very brief time, we did not add any security measures against impersonation.

The web app was deployed on our group-internal web server, which runs a standard LAMP stack (Linux operating system, Apache web server, MySQL database, and PHP). After some brief

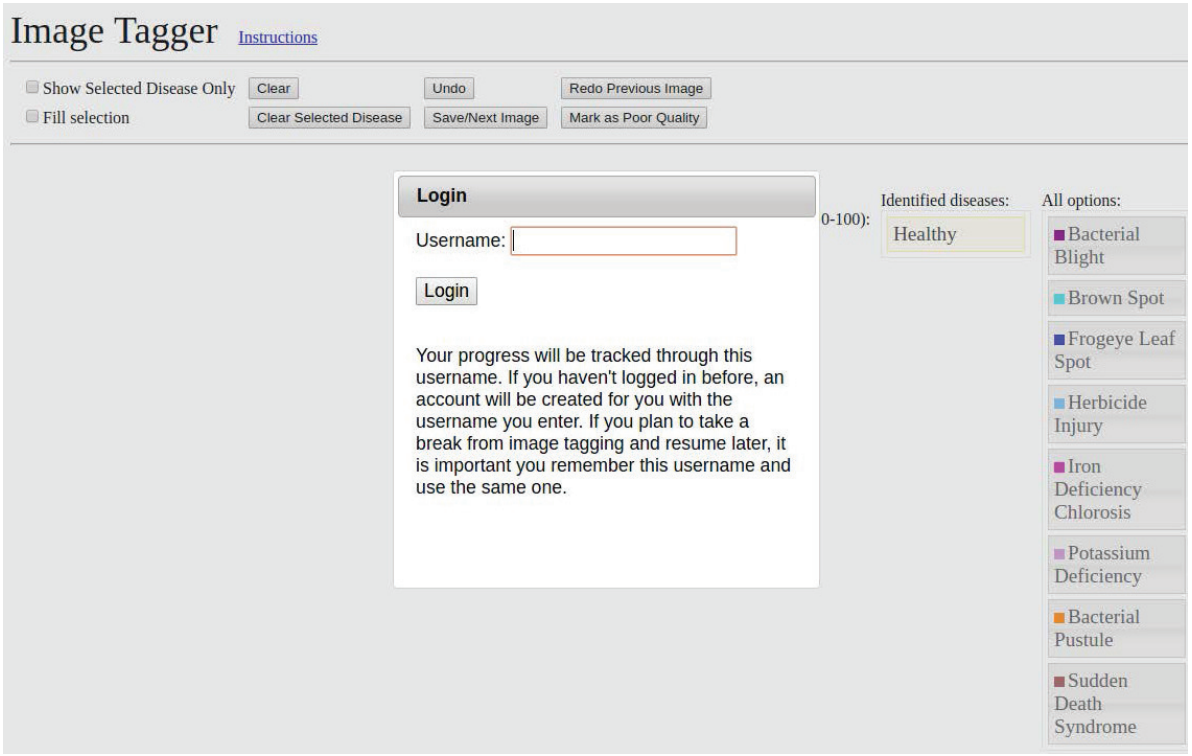



Figure 3.1 Design of the web app for expert-based foliar disease identification and severity estimation

testing, links to the application and instructions were distributed to experts to begin tagging the images. Tagging was conducted in multiple sessions during the course of one week (see Fig. S10).

The rater-based severity was calculated by marking the diseased part of the leaf and obtaining the ratio of the area (pixel count) of the marked region to the area of the entire leaf. This procedure was performed on a subset of 1000 random images extracted from the test set data (of 6576 images). Figure 3.9 presents examples of Grad-CAM and expert-rated leaf images for each disease and healthy cases.

## 3.2 Leaf Sample Collection and Data Generation

Efforts to develop the deep learning framework for the identification, classification and quantification of soybean stress began with the accumulation of images of stressed and healthy soybean leaflets in the field.


Figure 3.2 Illustration showing the tagging process

The labeled data needed for training were collected following a rigorous imaging protocol using a standard camera. The imaging platform (see Figure 3.3) comprised a flat rectangular surface covered with a dark cloth used to limit background noise. The platform was shaded by a large umbrella to ensure a consistent light source during imaging. Four leaves at a time were placed along the four corners of the platform, a color chart was placed in the center, and the leaves and chart were manually imaged using a digital camera (Canon EOS Rebel T5i, 18 megapixels). Using image processing techniques, the original images were segmented into four separate leaf images. The images were appropriately labeled as one of the eight different stress classes on the basis of the earlier field diagnoses. Each soybean stress class contained approximately 2000 leaf images with varying degrees of severity, whereas the healthy soybean class contained over 5000 images.

Over 25,000 labeled images (dataset available online) were collected to create a balanced dataset of leaflet images from healthy soybean plants and plants exhibiting eight different stresses (see Figure 3.4). Leaflet images were taken from plants in soybean fields across the state of Iowa, which is located in the United States. This dataset represents a diverse array of symptoms across

Figure 3.3 Leaf Imaging Platform

biotic (e.g., fungal and bacterial diseases) and abiotic (e.g., nutrient deficiency and chemical injury) stresses.

### 3.2.1 Data Collection

A total of eight different soybean stresses were selected for inclusion in the dataset, on the basis of their effects on yield loss in the state of Iowa. The eight soybean stresses included the following: bacterial blight, bacterial pustule, Septoria brown spot, sudden death syndrome (SDS), frog-eye leaf spot, herbicide injury, potassium deficiency, and iron deficiency chlorosis (IDC) (78). Healthy soybean leaflets were also collected to serve as a ground truth for the machine learning model for the successful differentiation of healthy and stressed leaves. First, various soybean fields in central Iowa associated with Iowa State University were scouted for the desired plant stresses. Entire plant samples were collected directly from the fields and taken to the Plant and Insect Diagnostic Clinic at Iowa State for official diagnosis by expert plant pathologists; for more information and online access, please follow this link to the online Plant and Insect Diagnostic Clinic. The exact locations of the sampled soybean plants were recorded at that time. After the stress identities were confirmed by the Plant and Insect Diagnostic Clinic, the desired fields were revisited. Individual

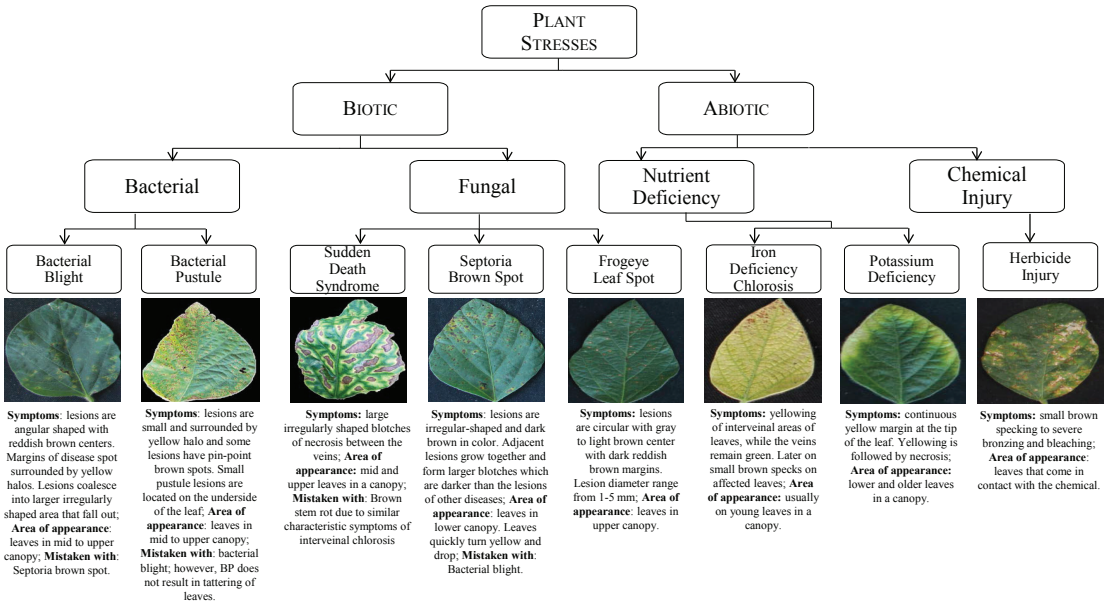



Figure 3.4 Schematic illustration of foliar plant stresses in soybean grouped into two major categories, biotic (bacterial and fungal) and abiotic (nutrient deficiency and chemical injury) stress. The images were used to develop the DCNN for the following eight stresses: bacterial blight (*Pseudomonas savastanoi* pv. *glycinea*), bacterial pustule (*Xanthomonas axonopodis* pv. *glycines*), sudden death syndrome (SDS, *Fusarium virguliforme*), Septoria brown spot (*Septoria glycines*), frogeye leaf spot (*Cercospora sojina*), IDC, potassium deficiency and herbicide injury. For each stress, information such as symptom descriptors, areas of appearance and most commonly mistaken stresses that exhibit similar symptoms are listed. These particular foliar stresses were chosen because of their large economic impact on agriculture and confounding symptoms.

soybean leaflets exposed to a range of different severity levels were then identified and collected manually through destructive sampling. Diseases such as frog-eye leaf spot, potassium deficiency, bacterial pustule and bacterial blight were present at low to medium intensity. The leaflets were placed into designated bags and transported to the imaging platform on-site.

### 3.2.2 Data Preparation and Generation

The dataset for training, validation and testing was prepared in the following manner: First, the images of the leaves were segmented out from the raw images and reshaped into images of pixel size 64X64 [(*height*) X (*width*)] to generate images for training the deep neural network efficiently. We used 4174 images for healthy leaves, 1511 images for bacterial blight, 1237 images for brown spot, 1096 images for frogeye spot, 1311 images for herbicide injury, 1834 images for IDC, 2182 images for potassium deficiency, 1634 images for bacterial pustule and 1228 images for SDS, i.e., a total of 16207 clean images. Refer to Figure 3.4 that show example images for each class, where Bacterial Blight Injury was designated as class 0, Brown Spot as class 1, Frogeye Leaf Spot as class 2, Herbicide Injury as class 4, Iron Deficiency Chlorosis as class 5, Potassium Deficiency Syndrome as class 6, Bacterial Pustule Injury as class 7, Sudden Death Syndrome as class 8 and finally healthy leaves as class 3.

A data augmentation scheme was adopted to increase the size of the dataset. To perform data augmentation, we used 1096 images for each of the disease classes and 2192 from the healthy class. The following augmentations were conducted: horizontal flip, vertical flip, 90 degree clockwise (CW) rotation, 180 degree CW rotation and 270 degree CW rotation, (see supplementary). We generated a total of 65760 images, which were then divided into training, validation and test sets in a 7:2:1 proportion. Refer to Figure 3.5 for an illustration of the data augmentation scheme.

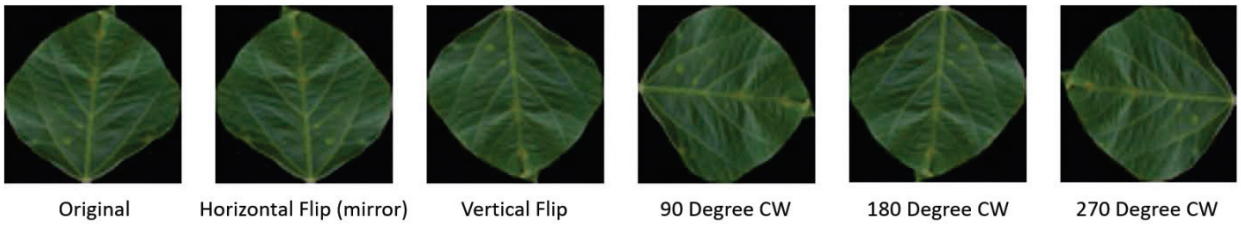



Figure 3.5 Illustration of data augmentation scheme

## 3.3 Deep CNN Model and Explanation Framework

We built a deep convolutional neural network- (DCNN-) based supervised classification framework to identify and classify stresses (Figure 2 (a) and (b)). DCNNs have shown an extraordinary ability (1, 2, 3, 4, 5, 79, 80) to efficiently extract complex features from images and function as a classification technique when provided with sufficient data (see Section 3.3.2). The exhibited accuracy is especially promising, given the multiplicity of similar and confounding symptoms between the stresses in single crop species (see Figure 1). We associate this classification ability with the hierarchical nature of this model (81), which is able to learn features of features from data without the time-consuming hand-crafting of features (see CHAPTER 2 Section 2.3). Upon classification inference, we use the Grad-CAM algorithm (69) to generate heat maps on a test leaf image that signifies the leaf region the DCNN model is focusing on to perform the classification.

### 3.3.1 Network Parameters

The DCNN architecture (shown in Figure 2) consists of 5 convolutional layers (128 feature maps of size $3X3$ for each layer), 4 pooling layers (down sampling by $2X2$ max-pooling), 4 batch normalization layers and 2 fully connected (FC) layers with 500 and 100 hidden units each, sequentially. Training was performed on a total of $53,265$ samples (with an additional $5,919$ validation samples), and testing was performed on $6,576$ samples. The learning rate was maintained at 0.1. The first convolutional layer maps the 3 bands ($RGB$) in the input image to 128 feature maps by using a $3X3$ kernel function. Subsequent max-pooling decreases the dimensions of the image. This step is performed by taking the maximum value in a window that is passed over the image. The stride here is the default stride, i.e., 2, which means that the window is moved 2 pixels at a time. This decrease in image dimension reduces computation complexity and time, thereby allowing the model to fit in the memory available from the graphics processing unit (GPU) (82).

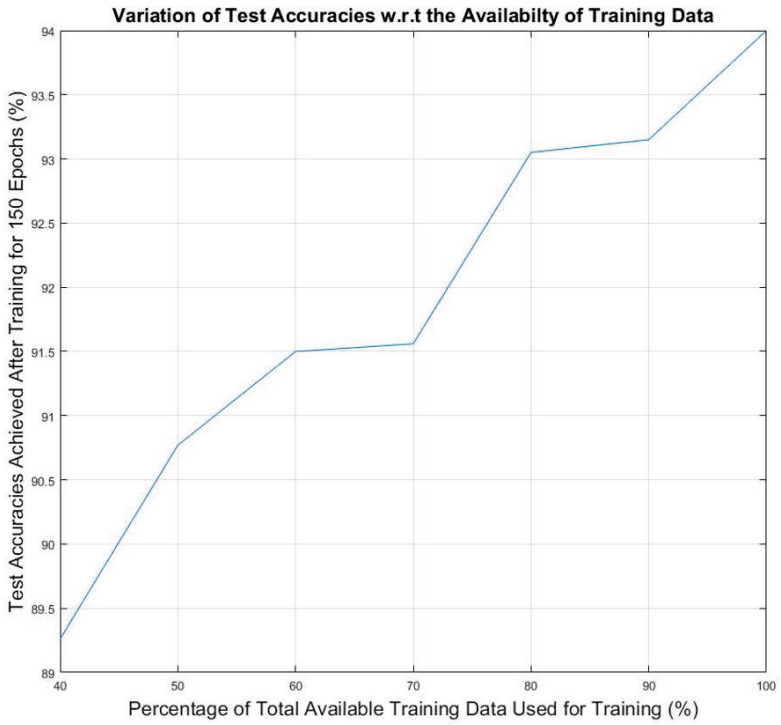
### 3.3.2 Training the DCNN

Our designed architecture initially had close to 3 million learning parameters, which was reduced to $\tilde{9}$00,000 while the same level of prediction accuracy was maintained. Although deep neural networks with a large number of learning parameters are very powerful architectures, overfitting becomes a severe problem in these cases. Large networks such as these are also slow to train, thus making it more difficult to address overfitting issues by combining the predictions of many large neural networks during the testing phase. Adding Dropout layers can usually solve such overfitting issues. The idea behind adding dropout layers is to randomly drop units along with their connections from the neural network during training (74). The percentage of dropouts used in our model is depicted in Figure 2 of the main text. The last FC layer after the final dropout layer gives the prediction for the class to which the input image belongs. After every convolutional layer, batch normalization was performed to remove internal covariate shift (63). The network was trained for approximately 100 epochs on the 53,265-image training set to reach the desired accuracy. The cross-entropy (categorical) loss (or cost) function along with the Adam optimizer (83) was used to minimize the error. Adam was chosen as the optimizer primarily because it requires minimal tuning of its internal hyper parameters. All other hyper parameters mentioned above have been cross-validated and chosen on the basis of repeated experiments to achieve the best possible prediction accuracy. Figure 3.6 shows how the prediction accuracy varies with the availability of training data, and suggests that performance has reached an asymptotic value.

## 3.4 Results

### 3.4.1 Disease Classification and Identification

While Figure 3 presents the qualitative results of deploying the trained DCNN for disease detection and classification, Figure 4 and Figure 5 detail the quantitative results over all test images. We found a high overall classification accuracy (94%) using a large and diverse dataset of unseen test examples (approximately 6000 images, i.e., approximately 600 examples per foliar stress).




| Percentage of Training Data Used for Training | Percentage Accuracy of Prediction Obtained for Test Data (after training for 150 epochs) |
|---|---|
| 40% | 89.26% |
| 50% | 90.77% |
| 60% | 91.50% |
| 70% | 91.56% |
| 80% | 93.05% |
| 90% | 93.15% |
| 100% | 94.00% |

Figure 3.6 Variation of test accuracy w.r.t. the increasing size of training data

The confusion matrix revealed that erroneous predictions were predominantly due to confounding disease symptoms that cause confusion even for expert raters (Figure 4). For example, the highest confusion (17.6% of bacterial pustule test images predicted as bacterial blight and 11.6% of bacterial blight test images predicted as bacterial pustule) occurred between bacterial blight and bacterial pustule; discriminating between these two diseases is challenging even for expert plant pathologists (84).

### 3.4.2 Symptom Explanation and Severity Quantification

In conventional disease-scouting scenarios, the ratings of even the same expert rater may change depending on various factors (intra-rater variability), such as illumination and human fatigue. Moreover, different human raters often tend to disagree (inter-rater variability), owing to the subjective quantification of the extent of symptoms expressed on a leaf (75, 85). In contrast, the trained

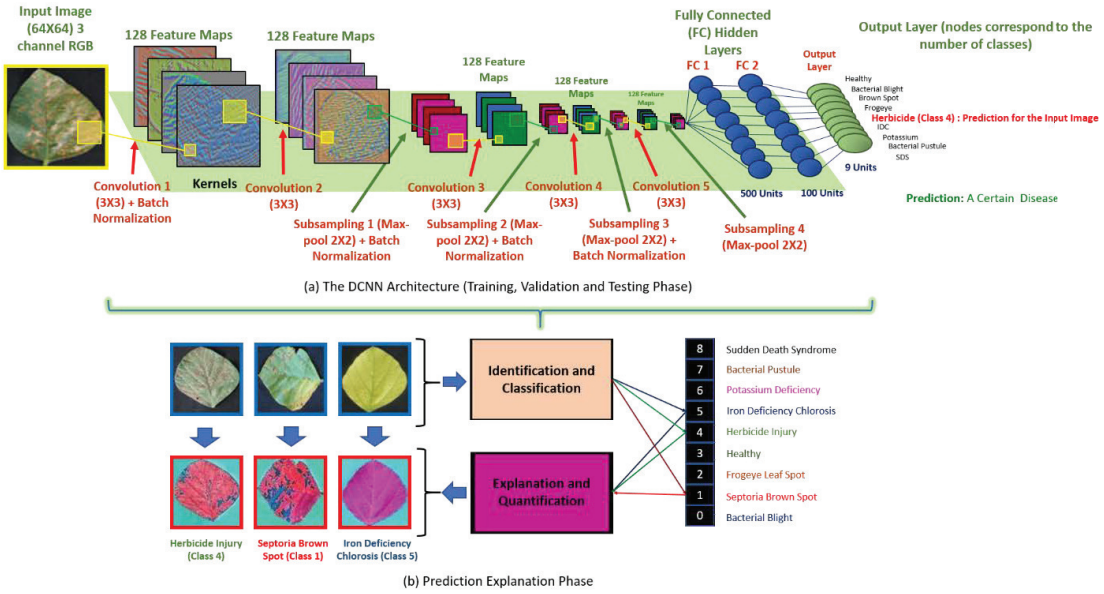



Figure 3.7 (a) DCNN architecture used. (b) Prediction Explanation Phase. The concept of gradient-based class activation mapping (Grad-CAM) was applied to automatically visualize image features used by the DCNN model when making a classification decision.

DCNN provides a consistent approach for severity estimation. Specifically, the spatial spread of the automatically identified symptoms allows for estimation of the severity of the classified disease in each leaflet. We computed the severity as the area fraction of identified symptoms and compared it with the severity estimated by an expert human rater. While the algorithm identified symptoms extremely precisely (at a pixel level), the expert human rater estimate was much more qualitative. This qualitative estimate was made by comparing the predicted severity rating to disease severity data collected from expert marking of symptoms challenging, primarily because of large inter-rater variability, as shown in Figure 4 (b). The decidedly more qualitative approach for the determination of disease spread by human experts resulted in severity values that were consistently larger than the precise machine learning-based annotations. We used a calibration process to map each human expert-annotated severity rating to a reference rating to make inter-rater comparison possible, thus rationally permitting the comparison of disease severity predictions between human raters and the trained model. This calibration process is detailed in section 3.10. Figure 3.11 (a, b,

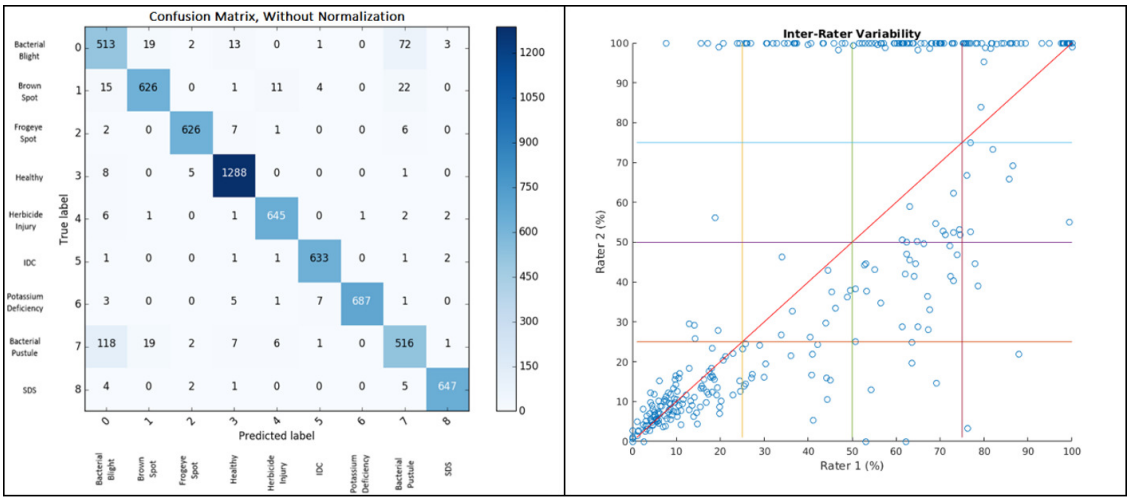



Figure 3.8 (a) This confusion matrix shows the stress classification results of the DCNN model for eight different stresses and healthy leaves. The overall classification accuracy of the model is 94%. The greatest confusion among stresses was found among bacterial blight, bacterial pustule, and Septoria brown spot. A reasonable explanation for these higher failure rates can be attributed to the similarities in symptom expression among these stresses.(b) is a scatter plot comparing the severity ratings of the same images between two raters for four stresses (IDC, SDS, potassium deficiency and Septoria brown spot) that were pooled and solid red line is the $45^o$ line. The results indicated high inter-rater variability between experienced raters, especially as the stress severity of leaf images increase.

c) shows a comparison of the machine learning-based ratings and human ratings based on a typical discretized severity scale (0-15%: resistant, 15-30%: moderately resistant, 30-45%: moderately susceptible, 45-75%: susceptible, and 75-100%: highly susceptible). Furthermore, these rating results demonstrated the efficacy of the DCNN-based severity estimation framework, which identifies the disease symptoms in a completely unsupervised manner. We observed that the few deviations in these results were primarily due to the low quality of the corresponding images, which exhibited shadows, low resolution and a lack of focus as detailed in section 3.4.2.2

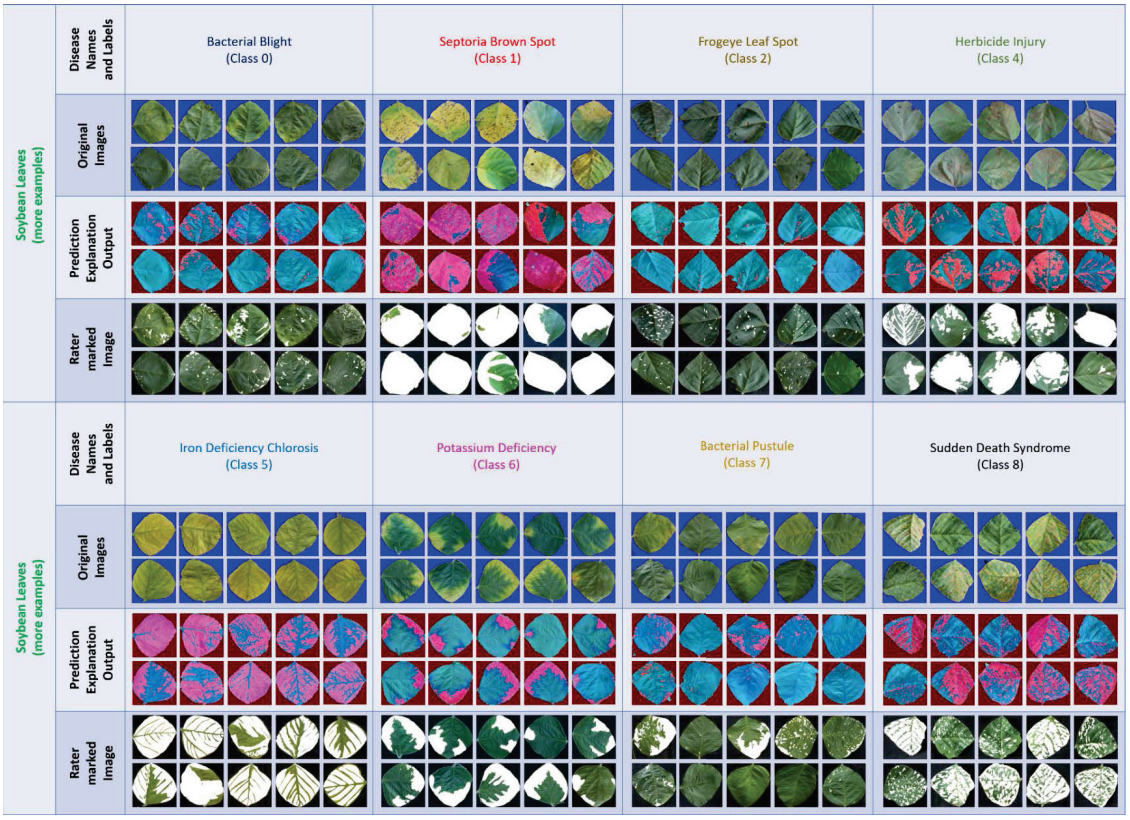



Figure 3.9 This table presents leaf image examples for each soybean stress identified and classified by the DCNN model. The Grad-CAM framework was applied to highlight regions of interest (symptoms) extracted by the DCNN model. These automatically extracted symptoms were compared against versions of images with symptoms that were marked manually by expert raters.

#### 3.4.2.1 Calibration Procedure

The machine learned severity, $\mathbf{r}^i_{m/c}$, performed quite well with many leaf images and was consistent with the user ratings, $\mathbf{r}^i_u$. However, in some cases, $\mathbf{r}^i_{m/c}$ provided a lower estimate of severity than the expert-based severity because human expert-based markings tended to be much coarser. In effect, typically all regions surrounding an affected region on a particular affected leaf are highlighted. However, the machine learning model via Grad-CAM is often too specific (at a pixel-level resolution) and highlights only the most important regions for a particular class to be activated. Therefore, a calibration process is required to achieve a uniform scale for machine learning-based severity and expert-based severity.

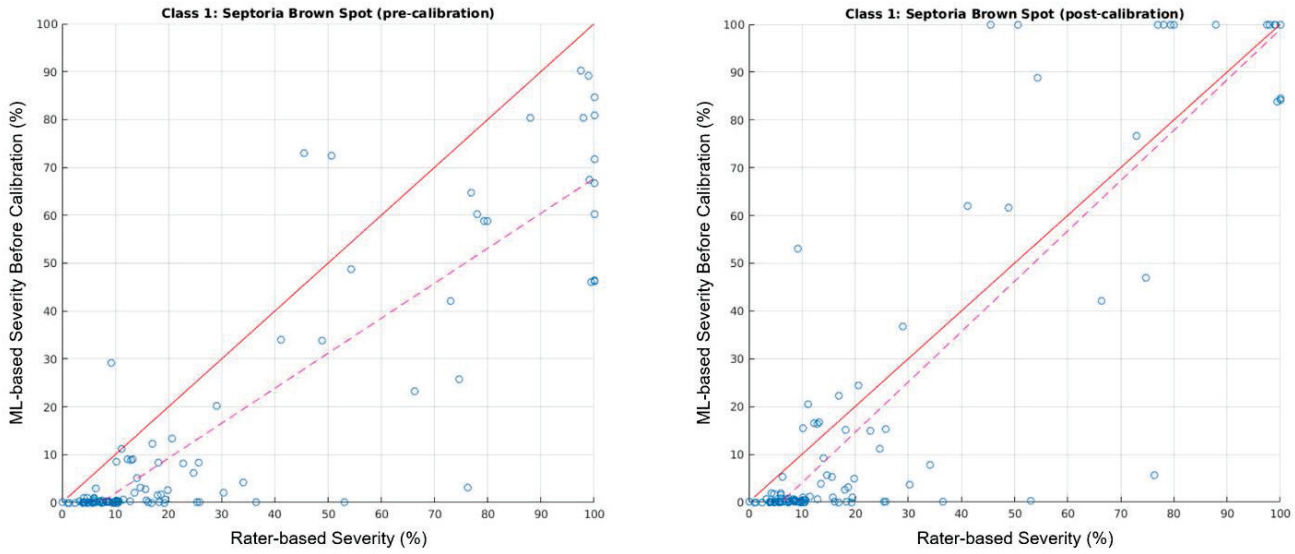



Figure 3.10 Rater-based severity versus ML severity scatter plots (pre-calibration: left and post-calibration: right) for class 1: Septoria brown spot. The solid red line is the reference 45-degree slope line and the dashed red line shows the linear fit applied to the points on the scatter plot in both cases.

We included disease classes 1 (Septoria brown spot), 5 (IDC) and 8 (SDS) in Figure 4 in the main paper as quantification example sets, because they demonstrate a large variation in severity across the entire range. We performed the calibration process for classes 1 and 5 while class 8 did not require calibration, because the results pertaining to class 8 already showed a promising trend without the need for additional calibration. We also aimed to keep the calibration model/process as simple as possible, using the expert-based severity ratings of only a few examples, so that the efficacy of the unsupervised machine learning-based severity rating process would be accurately demonstrated. The simple linear calibration process is detailed below.

The calibration was performed by initially randomly selecting $k(k << n)$ points with expert-based severity ratings from the test dataset (consisting of n data points) for a particular disease, such that they uniformly represented all levels in the discretized severity scale (**0-25%:** resistant, **25-50%:** moderately resistant, **50-75%:** moderately susceptible, and **75-100%:** susceptible). For only these k examples, we compared the severity ratings from the expert and machine learning models. We then calculated the slope as follows: using $\hat{\mathbf{r}} = \mathbf{r}_u^i$ for the selected points from the

dataset, the slope, $\overline{M}$, was calculated by fitting a linear model with the zero intercept for the scatter plot initially obtained between the expert severity rating and the machine-learning (ML) model severity rating. Thus, $\overline{M}$ is the slope of this linear model; with a value for $\overline{M}$, we obtain new machine severity rating values for the entire dataset:

$$\bar{r}^i_{m/c} = \overline{M} \odot r^i_{m/c} \tag{3.1}$$

The scatter plots of $\bar{r}^i_{m/c}$ versus $r^i_u$ depict the new location of the calibrated data points. The next step was to locate and calculate the number of points within the pre-defined bounds on the scatter plots, and then calculate the confusion matrices. These results indicated machine rating performance in comparison with the user rating, thus producing the plots shown in Figure 5 of the main text. The calibration process is shown for class 1, Septoria brown spot in the scatter plots of rater-based severity versus ML severity (pre- and post-calibration), as shown in Fig. S12.

The method discussed here is a valid calibration procedure, because Figure 4 (b) in the main text clearly suggests that calibration may be required even when human-annotated severity ratings are compared between two human raters.

#### 3.4.2.2 Failure Cases

Here we mention some of the failure cases that show up while running the framework:

- The presence of shadows and dark spots on the input image: The presence of shadows and dark spots in many examples (even on healthy leaves), posed a challenge for Grad-CAM in detecting the correct disease signatures (shadow regions are true disease identifiers in some cases), because the trained DCNN model sees the shadow and/or dark regions compared to yellow and brown regions as regions of low interest and thus tends to neglect those regions as untrue disease signatures. However, human raters do not make this mistake.

- The lack of (or unbalanced) focus when capturing the image: Blurry images did not serve as good examples for training or testing the neural network. The architecture in such cases

failed to detect proper disease signatures. Disease signatures for blurry (out of focus) images are also difficult for human raters.

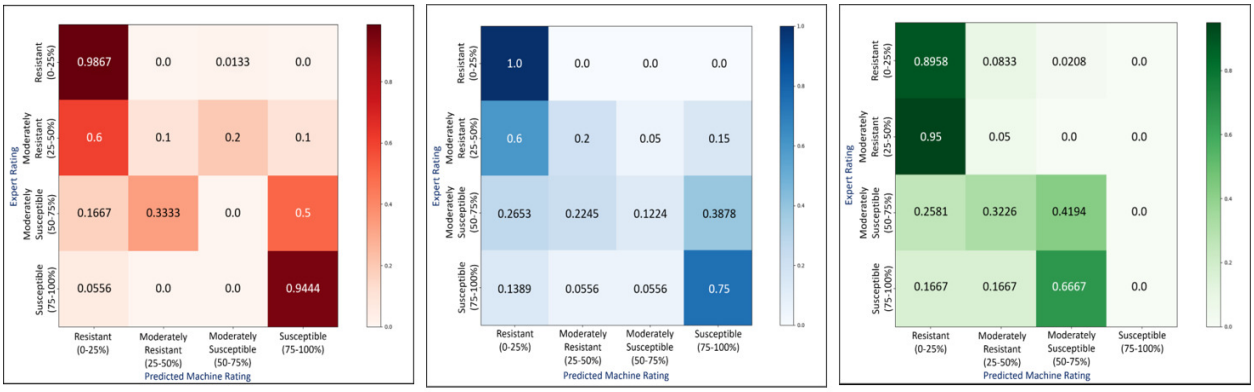
- The lack of resolution: To address GPU memory issues, the original high-resolution images were resized to low-resolution (64X64) images that were fed into the neural network. This resizing led to information loss and, in some cases, to the model and Grad-CAM failing to detect the correct disease signatures. The disease signatures of low-resolution images are also difficult for human raters to identify. In our case, high-resolution images were provided to the raters to identify correct disease signatures, whereas low-resolution versions of the same we fed to the neural network to alleviate GPU memory issues. Thus, the lack of resolution posed a problem in several cases when testing the DCNN architecture.

- Incorrect prediction by the CNN model: The presence of all or some of the previously mentioned three causes led to incorrect prediction by the neural network. When incorrect prediction occurred, Grad-CAM also failed to detect the correct disease signatures on the leaf.

An example for each of the mentioned failure cases is presented in Figure 3.12.

## 3.5 Discussions

The identification of human-interpretable visual cues provides users with a formal mechanism ensuring that predictions are useful (i.e., determining whether the visual cues are meaningful). Additionally, the availability of the visual cues allows for the identification of disease types and severity classes that are under-performing (those in which the visual cues do not match the expert-determined symptoms), thus potentially leading to more efficient retraining and focused data collection. Here, we emphasize that the identification of visual symptoms involves a completely unsupervised process that does not require any detailed rules (e.g., involving colors, sizes and shapes) to identify the symptomatic regions on a leaf; hence, this process is extremely scalable. Furthermore, the automated identification of visual cues could be used by plant pathologists to identify early symptoms of disease. In the context of plant stress phenotyping, four stages of the problem are




(a) (b) (c)

Figure 3.11 These figures (a, b, c) details the comparison between human and machine learning-based severity ratings for three previously mentioned stresses [(a) Septoria brown spot, (b) IDC, and (c) Sudden Death Syndrome]. The severity comparison using a standard discretized severity scale (0-15%: resistant, 15-30%: moderately resistant, 30-45%: moderately susceptible, 45-75%: susceptible, and 75-100%: highly susceptible) shows the success of the DCNN-based severity estimation framework to correctly quantify symptoms for these stresses. Classes such as resistance and highly susceptible tended to have high accuracy across the three stresses, whereas classes such as moderately susceptible were associated with greater confusion.

defined (86), namely, identification, classification, quantification and prediction (ICQP). In this paper, we provide a deep machine vision-based solution to the first three stages. The approach presented here is widely applicable to precision and digital agriculture and allows for more precise and timely phenotyping of stresses in real time. We envision that this approach could be easily extended beyond plant diseases (i.e., to animal and human diseases) and other imaging modalities (hyper-spectral) and scales (ground and air), thereby leading to more sustainable agriculture, food production, and health-care.

### 3.5.1 Transfer Learning Capability

Well-trained DCNNs learn to generalize features rather than memorize patterns (87). We leveraged this characteristic to explore whether a model trained specifically for soybean diseases could make accurate predictions for other plant species with the same diseases. This capability for

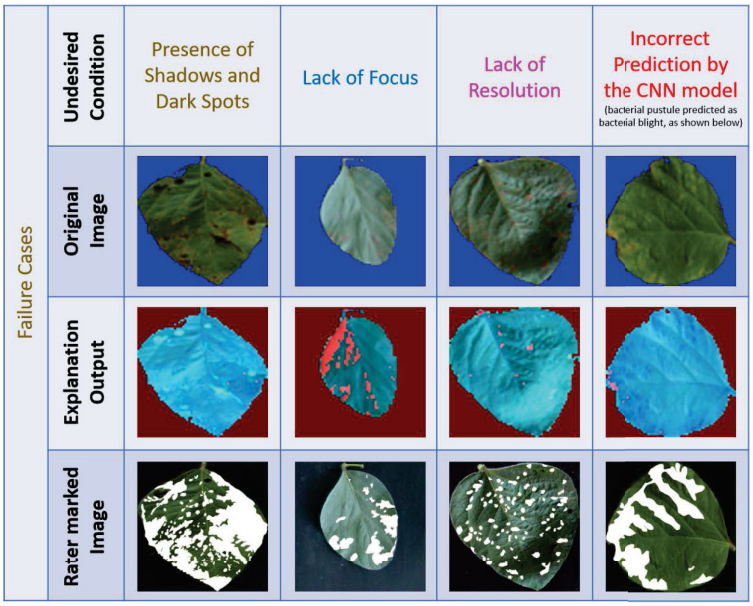



Figure 3.12 Examples in which Grad-CAM failed to detect proper disease signatures

transfer learning (88) was demonstrated with several non-soybean leaf image examples (frog-eye leaf spot in apple, IDC in cucurbits and potassium deficiency in oilseed rape). The examples shown in Figure 6 demonstrates that the algorithm was effective in accurately identifying diseases and their symptoms in non-soybean plants.

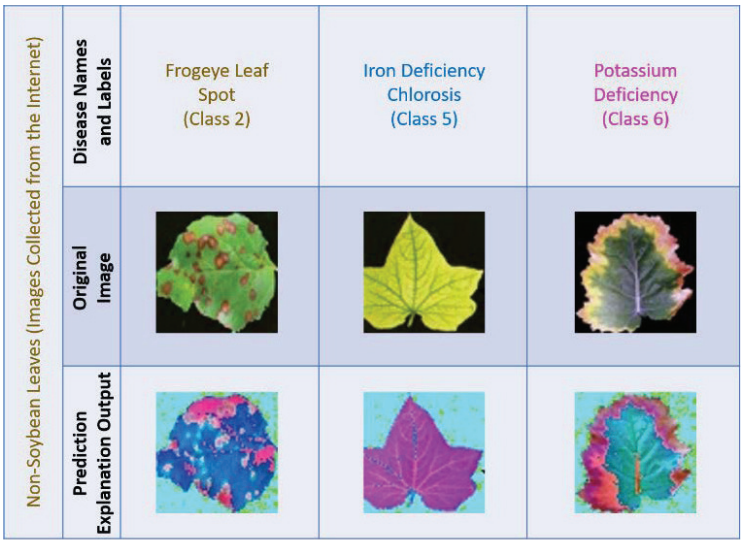



Figure 3.13 This table illustrates the ability of the Grad-CAM framework to accurately identify the same stress symptoms used in training the DCNN model to non–soybean crops, such as frogeye leaf spots in apple leaves, IDC in cucurbits and potassium deficiency in oilseed rape.

# CHAPTER 4. EXPLAINABLE AND APPLIED 3D DEEP LEARNING: EARLY DETECTION OF COMBUSTION INSTABILITIES FROM HIGH-SPEED VIDEO

Understanding flame dynamics and combustion instability is a complex problem involving different non-linearities. Combustion instability has several detrimental effects on flight-propulsion dynamics and structural integrity of gas turbines and any such spaces where combustion takes places internally, primarily in internal combustion engines. The description of coherent features of fluid flow in such cases is essential to our understanding of the flame dynamics and propagation processes.

In the present day, flame dynamics is an important topic of study owing to the complexities involved and also because of its immense applications. The non-linear and chaotic behaviour of flames and the physics associated with it arouses significant interest for many researchers. Flame dynamics is a result of coupling between turbulence, combustion and acoustics which lead to combustion instabilities. Such fluid flows are infinite dimensional systems governed by nonlinear partial differential equations. Even so, the essential features of their dynamical responses can be effectively approximated by models of lower complexity. These models utilize the concept of coherent structures. Coherent structures are organized fluid elements that, along with dynamic processes, are primarily responsible for most of the energy and momentum transfer in the flow. These structures are those whose generation mechanisms vary from system to system, and cause velocity oscillations and flame shape oscillations by curling and stretching. A few popular physics based methods used for detecting these coherent structures are Proper Orthogonal Decomposition (POD) (89) and Dynamic Mode Decomposition (DMD) (90, 91), which utilises tools from spectral theory to derive spatial coherent structures. But these methods mostly do so in a supervised manner and utilises handcrafting of features.

Recently, deep learning models have been shown to outperform other state of the art techniques in handling and analyzing large dimensional data (both spatial and temporal), by learning the hierarchical features to perform various tasks such as, classification and bulk structure detection given a large corpus of 2-dimensional (2D) data or images. As an extension, embedding of 3-dimensional (3D) spatiotemporal data (where the data have spatial features evolving over time) has been performed using a 3D convolutional neural network (3D CNN) framework. In this work, we apply for the first time such a 3D CNN architecture for early detection of combustion flame instability using hi-speed flame videos. We demonstrate the performance of our proposed framework on an experimental data set of hi-speed (i.e., sampling at 3 KHz) video consisting of flame images collected from a laboratory scale swirl-stabilized combustor. Such early detection of combustion instability can eventually enable health monitoring and enhance fuel efficiency of an engine or power generation system that uses a combustor. The main contribution of the work is development of a 3D CNN model that is pre-trained by the encoder part of a 3D convolutional autoencoder that outperforms the recent results reported using 2D deep learning frameworks (that are unable to leverage the temporal correlations among the consecutive frames) on the same data set. For training the 3D network, multiple 2D frames are stacked into short temporal segments and the performance of the model is evaluated in detecting regions of stability, instability and most importantly regions of evolving flame intermittencies that is crucial for an early warning system.

In general, the model development and analysis presented in this work opens up the door for leveraging 3D CNN models for prognostics and health monitoring of human-engineered systems using large volume spatiotemporal data. In this work, We propose a deep learning-based tool that has enormous potential for preventing catastrophic failure in engines by early detection of combustion instability. Control modules can leverage such information to enable recovery of the combustion process at least to a gracefully degraded condition. Such diagnostics tools are of particular interest because of the current focus on reduction of fossil fuel use in jet engines and power plants. However, the equivalent mixtures obtained by reducing the fuel-to-air ratio in the combustors have been found to result in combustion instabilities (92). Among the common effects

of instabilities that hurt the health and reliability of the engines are the blow-out and excessive flame heating that cracks the wall of the chamber. Therefore, the safety of the customers and operators of such systems become compromised.

As sensors such as pressure or chemiluminescence may not be able to detect sufficiently early, recent research studies have leveraged high-speed flame video and sparse DMD has been applied to detect stability and instability in the flame images by essentially extracting the primary modes of the evolving flame (93). Also, a neural-symbolic framework was explored for extracting salient features from multi-modal sensors data that uses labeled data of stable and unstable frames. The framework fuses the features from the images with the information provided by the pressure sensor (94). An end-to-end (from image to image) type 2-dimensional convolutional selective autoencoder (2D CAE) was designed to jointly learn the features that encode the information in the stable frames as well as the coherent structure information present in the unstable frames (95). The 2D CAE network was successful at revealing the intermittent instabilities, however, due to the inability of capturing temporal characteristics, the performance of such models was sub-optimal especially in detecting intermittent instability indicators before the flame becomes completely unstable. To capture such dynamics of the frames, a 3D convolutional neural network (3D CNN) was trained.

## 4.1 Problem Formulation and Experimental Setup

### 4.1.1 Problem Statement

Thermo-acoustic instability relates to the excitement of acoustics in a resonator with heat release rate fluctuations as the amplifier and source of such acoustics. The heat release rate fluctuations can be positively coupled to the pressure fluctuations through various mechanisms. Such mechanisms in general, are: 1. Velocity coupling and 2. Fuel concentration fluctuations. Velocity coupling is a broad term for different mechanisms, that involve both hydrodynamic and flame response to the former.

A prominent sustainer of combustion instabilities, especially in turbulent combustors is flame vortex interaction (90). In fact, a number of studies on bluff body and rearward facing step

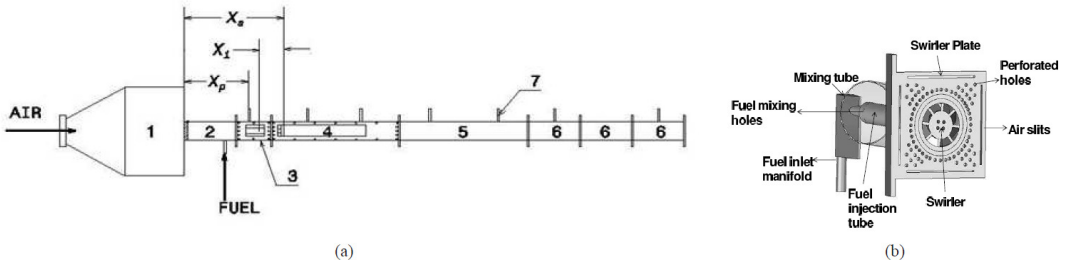



Figure 4.1 (a):Schematic of the experimental setup. 1 - settling chamber, 2 - inlet duct, 3 - IOAM, 4 - test section, 5 - big extension duct, 6 - small extension ducts, 7 - pressure transducers, $X_s$ - swirler location measured downstream from settling chamber exit, $X_p$ - transducer port location measured downstream from settling chamber exit, $X_i$ - fuel injection location measured upstream from swirler exit, (b):Swirler assembly used in the combustor.

combustors have highlighted in a qualitative manner (visual inspection) over a pressure cycle, the formation, evolution and interaction with flames (95, 96). Reports in (97) have shown the onset of combustion instability as "lock-on" of the system acoustics to Karman vortex shedding mode, thus clearly marking such structures as the drivers of combustion instability. It must be noted that in any high $Re$ (Reynold's Number) turbulent system, vortices of varying scales and intensity are present all the time, which however do not drive combustion instability. This is due to them not being coherent vortices as opposed to the vortices resulting from hydrodynamic instabilities. Coherent motions cause "in phase" or "uniform modulation" of the flames, thus causing large scale heat release rate fluctuations and consequently, combustion instability.

Detection of these coherent structures has been conventionally through visual inspection on flame images or through linear splitting techniques like POD (98). The former is subjective and the latter though well tested, is applicable to only statistically stationary data. There are also concerns on POD modes being entirely physical in nature, although POD naturally identifies field data having high correlation values - a definitive property of coherent motion.

### 4.1.2 Experimental Setup and Description

The swirl combustor test bed used in this study has a swirler of diameter 30 mm with 60 degree vane angles, thus yielding a geometric swirl number of 1.28. Air to the combustor is fed through a settling chamber of diameter 280 mm with a sudden contraction leading to a square cross section of side 60 mm. This provides an area ratio of around 17, which thus acts as an acoustically open condition at the contraction. A mesh and honeycomb are mounted in immediate downstream of the contraction to provide uniform flow to the swirler. The combustor, shown in figure **??**(a) consists of an inlet section of length 200 mm, an inlet optical access module (IOAM) of length 100 mm to provide optical access to the fuel tube, a primary combustion chamber of length 370 mm, and secondary duct of the same length. Extension ducts of the same cross section are added to provide length flexibility. The overall length of the constant area ducts was chosen to be 1340 mm.

The fuel injection is done by injecting it coaxially with the air in a fuel injection tube with slots on the surface as shown in Figure 4.1(b). The fuel injection tube is coaxial to a mixing tube which has the same diameter as that of the swirler. The bypass air that does not enter the mixing tube passes through slots on the swirl plate. The slots on the fuel injection tube are drilled at designated distance upstream of the swirler. The larger this distance, more fuel mixes with the primary air in the mixing tube thus leading to more premixedness. Two upstream distances of $X_1 = 90mm$ and $X_2 = 120mm$ were chosen for this work. The upstream distance of 120 mm provides for full premixing of the fuel with the air. The 90 mm upstream injection case causes partial premixing of the fuel with air. The images were acquired at 3 kHz using Photron High speed star with a spatial resolution of $1024 \times 1024$ pixels.

## 4.2 Dataset Generation

The training inputs of the network are composed of short time stacks of the 2D image frames that explicitly model the dynamics of neighboring dependent frames. An example of the time evolution of the stable and unstable frames is shown in Figure 4.4. The frames shown to be stacked to form the 3D super-voxels that motivate the implementation of our architecture. There were

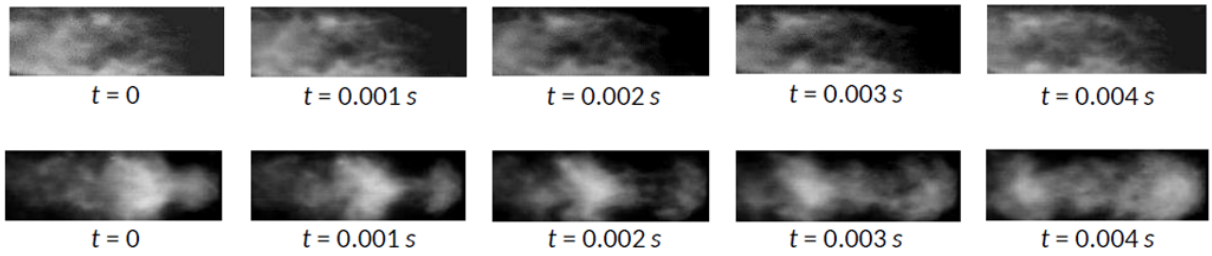



Figure 4.2 Combustion images used for flame stability analysis, captured at $3000 fps$ $(frames/second)$ *i.e.* $3kHz$; Top: greyscale images at $Re = 7,971$ and full premixing for a fuel flow rate of 0.495 g/s, bottom: greyscale images at $Re = 15,942$ and full premixing for a fuel flow rate of 0.495 g/s. *(Image Source: (99))*

63,000 frames available for training the images, out of which 35,000 frames were taken from the stable region and 28,000 frames were collected for the unstable region.

The 3D super-voxel data were generated using the following scheme: Each of the original 64 X 64 frames were first resized to 16 X 16 smaller frames. This dimensionality reduction is purely based on convenience and ability to fit through our available compute nodes due to the curse of dimensionality associated with 3D network. 32 such frames were then stacked together to generate the 3D data cubes (voxels), with each voxel being of size (32, (16, 16)). The training data size was 7592 3D super voxels while valiation was done on 2531 super voxels.

## 4.3 Network Architecture

The network in Figure 2 is made up of a series of convolution and maxpooling layers whose feature maps perform the detection of image and feature edges, shapes and objects. Convolutions are initiated by choosing appropriate kernel sizes and numbers that are shifted and convolved on the input images, following which, the reLU activation[5] is applied for learning the joint kernels in local neighborhoods that are useful for describing the explanatory features at that layer. Maxpooling layers are included to learn the scale invariance of the input images. The fully connected layers

| Premixing | FFR (g/s) | Re | Ground truth |
|---|---|---|---|
| Partial | 0.495 | 7,971 | Stable |
| | | 15,942 | Unstable |
| $(X_1 = 90mm)$ | 0.308 | 10,628 | Unstable |
| | 0.66 | | Stable |
| Full | 0.495 | 7,971 | Stable |
| | | 15,942 | Unstable |
| $(X_2 = 120mm)$ | 0.308 | 10,628 | Unstable |
| | 0.66 | | Stable |
| | 0.083 | 1,771 | Relatively stable |

Figure 4.3 Description of operating conditions along with respective ground truth (stable or unstable) for hi-speed image data collection.

embed the high dimensions in a low dimension space, activates ensure that the orientations do not necessarily hurt the network performance and produces the classification of the network at the coding layer with a softmax activation (39).

Network training is completed by back-propagating the error in a reverse direction to the activation. In that process, the weights are modified by using some standard algorithm such as the information-theoretic binary cross entropy function (39) which examines the level of filter achieved at each run of the algorithm. After various inferences using different network structure, we found a short time segment of 32 frames suitable for better detection of the intermittent unstable frames before the onset of complete instability.

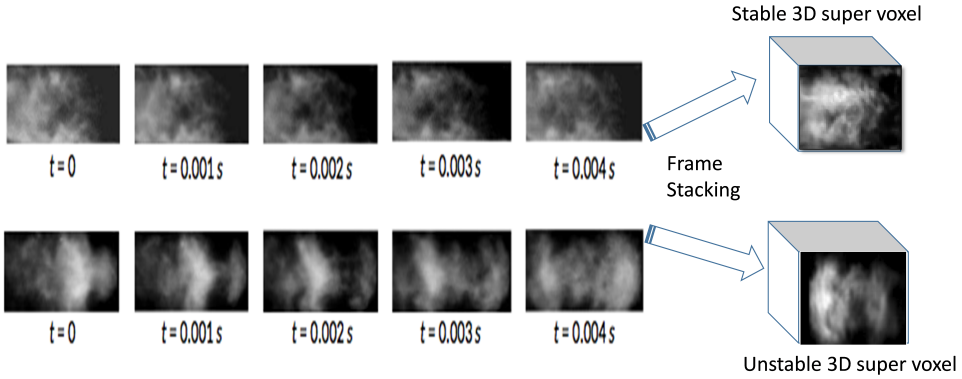



Figure 4.4 Examples of stable and unstable flame frames in a typical temporal evolution and their respective 3D super-voxels.

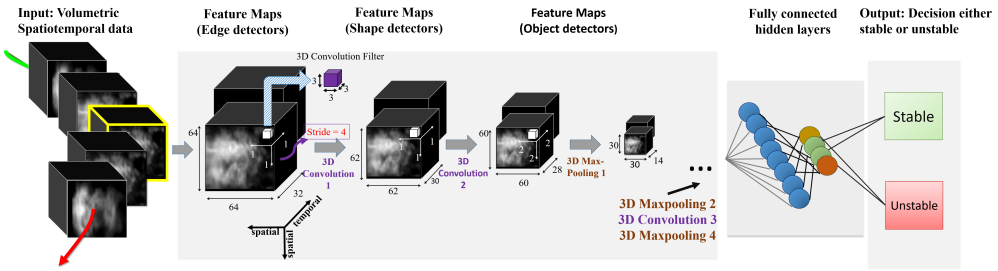



Figure 4.5 3D CNN architecture consisting of example image snapshots, hierarchical layers and model parameters that were trained to learn the suitable features from frames in the stable and unstable frames.

## 4.4 Results

### 4.4.1 Results from the 3D CNN model

Inference is done using unseen test videos (of 21,000 frames each) from various experiments with different air-to-fuel ration protocols such as: Protocols $500_{40to28}$, $500_{40to30}$ and $500to600_{40}$ where the digits are the air flow rate (in litres/minute) and the subscripts denote the fuel flow rate (also in litres/minute). The protocols model flame transition from stable to unstable regions in compliance with increasing the relative air-to-fuel ratio since the air flow has been reduced in the experiments. For each of the test protocols, there were 21,856 sequential frames of 2D flame

images available, which were then stacked into 683 3D super voxels, following the similar stacking principle as for the training and validation data sets. The machine learning tool described by the network in Figure 4.5 is examined to determine the dynamics that take place in between the stable and unstable frames with the goal of determining the instabilities present stable frames as well as the onset of instability. The results in Figure 4.6 show the instability levels against the 3D frame numbers.

### 4.4.2 Comparing the 3D-CNN based results with a 2D-based method, 2D CAE

From Figure 3, we note that the 3D CNN framework is able to efficiently detect onset of instability (termed as intermittency) in the flame flow marked by the peaks in the plots (encircles in red). These regions could not be detected earlier using the 2D based methods reported in literature. Thus, 3D CNN proves to be more effective in detecting these crucial points thus effectively analyzing the flame flow in a spatiotemporal manner. Apart from detecting the intermittencies efficiently, the framework can also predict the regions of stability and total instability effectively.

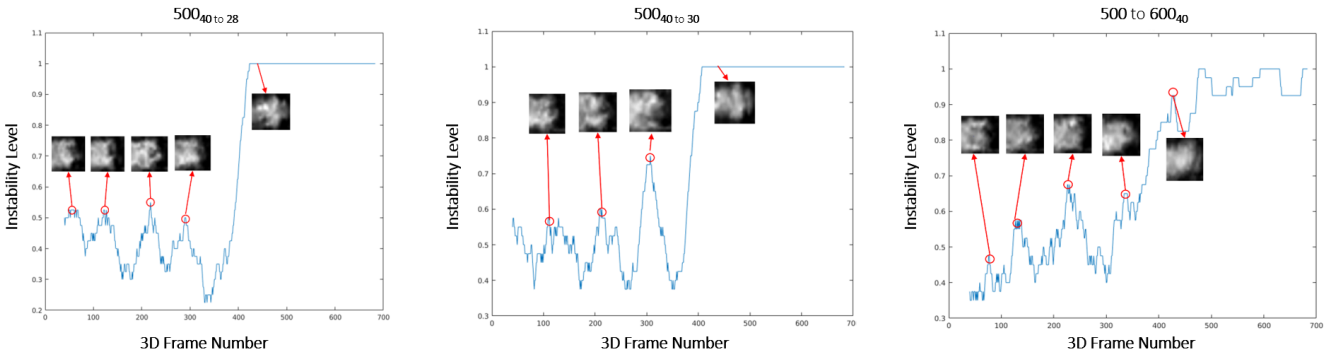



Figure 4.6 3D CNN Results for the 3 considered protocols with example frames from each of the unstable intermittencies in the insets.

**Generating the Instability Level Metric** The 3D-CNN architecture classified the voxels into either stable or unstable labels. From this classification, binary labels, 0 and 1 were assigned

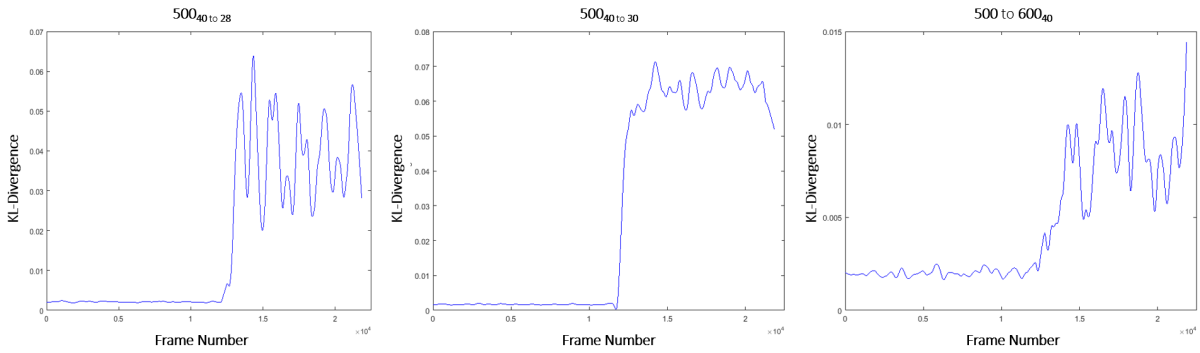



Figure 4.7 2D CAE Results for the 3 considered protocols.

to the stable and unstable outputs respectively. Then a 20-point moving average was performed to generae the Instability level metric as shown in Figure 4.6.

For the 2D CAE, the instability level metric is a metric based on the Kullback-Liebler (KL) divergence (100). This is chosen to measure the distance of the results from the image frames in each transition protocol from the expected result of a stable flame frame. This yields a KL distance, for each image frame. This physically corresponds to taking the distance of each image from the reference of the stable flame.

### 4.4.3 Explaining the results with 3D CAE

The frame (voxel) number for which the 3D CNN generated the peaks in the instability measure as seen in Figure 4.6 were noted and the same test protocol data were run through a Convolutional Autoencoder (CAE) framework. These frame numbers are: 57, 123, 218, 290 and 430 for the $500_{40to28}$ protocol, 110, 213, 307 and 418 for the $500_{40to30}$ protocol and 77, 132, 227, 339 and 430 for the $500to600_{40}$ protocol respectively. For the The results from the CAE framework showed that coherent structures detected by it for those particular frame numbers truly showed regions of high instability, thereby validating our approach that the 3D CNN was indeed able to learn and capture the underlying physics behind the induced acoustics in the flame which led to formation of such instabilities in the first place. Figure 4.8 shows how the CAE is able to filter out and consequently

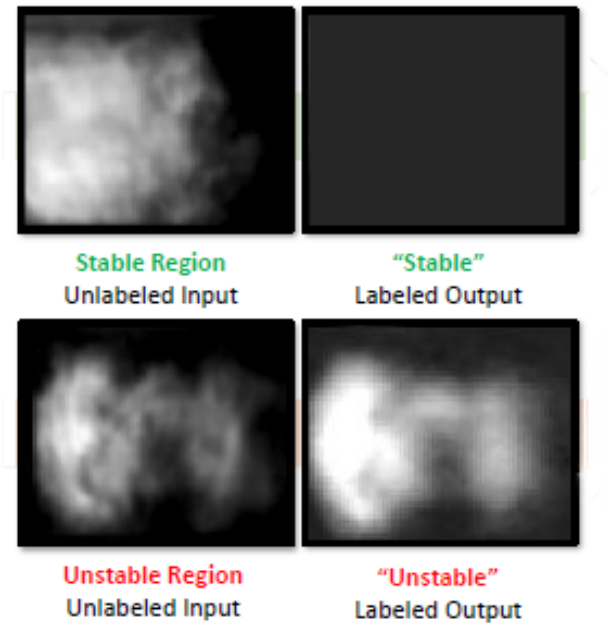



Figure 4.8 Illustration of CAE's ability to segregate regions of stability from instabilities from one another.

segregate regions of stabilities from regions of instabilities. Corresponding to the 3D Frame number designated as unstable by the 3D-CNN model, the images generated by the 3D CAE are included in Figure 4.6, as pointed out respective to their instability peaks.

# CHAPTER 5. SUMMARY AND CONCLUSION

In this chapter some conclusions are drawn from the works presented in CHAPTERs 3 and 4.

From both CHAPTERS 3 and 4, we have seen that the application of machine learning, and more specifically deep learning methods holds a vast potential in solving real world problems with inherent complexity, of which two have been detailed in these two earlier chapters. Deep Neural Networks have been shown to be capable enough of modeling complicated non-linearities in data and serves as a powerful tool in performing classification and object feature detection tasks (as demonstrated in chapter 3, where it is able to efficiently perform a 9-class classification of several soybean stresses from 2D RGB image data and in chapter 4, where it is able to perform the binary classification of stable and unstable flame zones from generated 3D spatiotemporal data with almost 100% accuracy (99.96% to be exact). A noteworthy point here is that through the use of purely Deep Learning based methods, we are able to extract spatial as well as temporal information from data (refer Chapter 4, where analysis is done upon the 3D voxels of flame images, which are basically cubes of spatiotemporal information). As is demonstrated in Chapter 4, a 3D based method like 3D-CNN is able to capture spatiotemporal information while 2D based methods (2D CAE) clearly fails to do so or under-performs to a significant extent. This leads us to state that capturing both spatial and temporal information from data where there is time-evolution involved is crucial and our method, 3D-CNN is able to do so quite efficiently.

Furthermore, in recent times, the scientific community deals with machine learning models as generic black-box models and consequently find them hard to trust. We address this issue by introducing an explanation framework for each of the 2D and 3D cases.

For the 2D case, we introduce Grad-CAM (Gradient weighted Class Activation Mapping) to locate those most important regions of the leaf image for which the CNN framework makes it's predictions and assigns the classes to each leaf image. We then further quantify this measure by

introducing a severity framework based on Grad-CAM. This leads us to calculate the percentage severity of each leaf image as well. The novelty of this approach lies in the fact that this is accomplished in a completely unsupervised manner (i.e., without the need for explicit labeling).

For the 3D case, we introduce the Convolutional Auto-encoder (CAE) to extract coherent structures for the 3D voxels for which the 3D CNN makes it's predictions. The CAE acts as a filter which masks the stable regions (i.e., does not produce any output image) while producing output images for the unstable flame zones only. The predicted stable and unstable zones match with that of the CAE outputs thereby verifying the correctness of our approach.

**Scopes for Improvement and Potential Future Research Directions**

- In regards to the plant phenotyping work, further work includes, but is not limited to - 1. developing a mobile platform based on the developed explainable DCNN framework, 2. Taking into account the cases where the framework failed to achieve optimal performance, 3. Deploy the further modified and optimal framework and achieve multi-sensor fusion to automate a ground-air-satellite based system to achieve our target goal, which is to fully and efficiently automate plant phenotyping.

- In regards to the combustion instability detection work, future research direction include, but is not limited to - 1. Perform experiments locally to gather data specifically targeted at instability detection and monitoring problems (this is currently under way), 2. Develop a deployable monitoring framework that can be bilt into mobile platforms to carry out fast and precise on-site flame instability detections.

# REFERENCES

[1] Esteva, A., Kuprel, B., Novoa, R. A., Ko, Justin, Swetter, S. M., Blau, H. M. and Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, *542*(7639), 115–118.

[2] Yamins, D. LK. and DiCarlo, J. J. (2016). Using goal-driven deep learning models to understand sensory cortex. *Nature Neuroscience*, *19*(3), 356.

[3] Alipanahi, B., Delong, A., Weirauch, M. T. and Frey, B. J. (2015). Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. *Nature Biotechnology*, *33*(8), 831–838.

[4] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G. and others (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.

[5] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, Marc and others (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489.

[6] Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*(2), 91–110.

[7] Dalal N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. *Computer Vision and Pattern Recognition*,

[8] Ojala T. et al (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, *24*(7), 971-987.

[9] Lazebnik. et al (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *Computer Vision and Pattern Recognition*, 2169–2178.

[10] Felzenszwalb F. P., Girshick, B. R., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern. Anal. Mach. Intell.*, *32*(9), 1627–1645.

[11] Fergus R., Perona, P., and Zisserman, A. (2003). Object class recognition by unsupervised scale-invariant learning. *IEEE Conference on Computer Vision and Pattern Recognition*, *2*, 264–271.

[12] Blei, D., Ng, A., and Jordan, M. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

[13] Hofmann, T. (2001). Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 2001.

[14] Adams, P. R., Wallach, M. H., and Ghahramani, Z. (2010). Learning the structure of deep sparse graphical models. *Journal of machine Learning Research - Proceedings Track*, *9*, 1–8.

[15] Zhu, L. L., Chen, Y., Yuille, A. and Freeman, W. (2010). Latent hierarchical structural learning for object detection. *IEEE Conference on Computer Vision and Pattern Recognition*, *0*, 1062–1069.

[16] Zhu, L., Chen, Y., Torralba, A., Freeman, W. and Yuille, A. (2010). Part and appearance sharing: Recursive compositional models for multi-view. *Computer Vision and Pattern Recognition (CVPR)*, 1919–1926.

[17] Chen, Y., Zhu, L. and Yuille, L. A. (2010). Active mask hierarchies for object detection. *ECCV, Lecture Notes in Computer Science*, *6315*, 43–56.

[18] Blaschko B. M. and Lampert H. C. (2008). Learning to localize objects with structured output regression *ECCV*,

[19] Schnitzspan, P., Fritz, M., Roth, S. and Schiele B. (2009). Discriminative structure learning of hierarchical representations for object detection *Computer Vision and Pattern Recognition (CVPR)*, 2238–2245.

[20] Wu, N. Y., Si, Z., Gong, H. and chun Zhu, S. (2008). Active basis for modeling, learning and recognizing deformable templates.

[21] Desai, C., Ramanan, D. and Fowlkes, C. (2011). Discriminative models for multi-class object layout. *International Journal of Computer Vision*,

[22] Torralba, A., Murphy, P. K. and Freeman T. W. (2005). Contextual models for object detection using boosted random fields. *Advances in Neural Information Processing Systems (NIPS) 17, MIT Press*, 1401–1408.

[23] Murphy, K., Torralba, A. and Freeman, T. W. (2003). Using the forest to see the trees: A graphical model relating features, objects, and scenes,

[24] Galleguillos, C., Rabinovich, A. and Belongie, S. (2008). Object categorization using co-occurrence, location and appearance. *Computer Vision and Pattern Recognition (CVPR)*, 1–8.

[25] He, X., Zemel, R. and Carreira-Perpindn, M. (2004). Multiscale conditional random fields for image labeling. *Computer Vision and Pattern Recognition (CVPR)*, *2*, 695–702.

[26] Hoiem, D., Efros, A. A. and Hebert, M. (2008). Putting objects in perspective. *International Journal of Computer Vision*, *80*(1), 3–15.

[27] Baur, R., Efros, A. and Hebert, M. (2008). Statistics of 3d object locations in images.

[28] Kumar, S. and Hebert, M. (2005). A hierarchical field framework for unified context-based classification. *Tenth IEEE International Conference on Computer Vision (ICCV)*, *2*, 1284–1291.

[29] Goodfellow, J. I., Le, V. Q., Saxe, M. A., Lee, H. and Ng, Y. A. (2009). Measuring invariances in deep networks.

[30] Zeiler, D. M. and Fergus, R. (2013). Visualizing and understanding convolutional networks. *CoRR*,

[31] LeCun, Y., Bottou, L., Bengio, Y. and Haner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

[32] Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., Montral, D. U. and Qubec, M. (2007). Greedy layer-wise training of deep networks. *NIPS, MIT Press*,

[33] Hinton, E. G. and Osindero, S. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*.

[34] Sermanet, P., Chintala, S. and LeCun, Y. (2012). Convolutional neural networks applied to house numbers digit classication. *21st International Conference on Pattern recognition (ICPR)*, 3288–3291.

[35] D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classication. *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3642–3649.

[36] Wan, L., Zeiler, M., Zhang, S., Lecun, Y. and Fergus, R. (2013). Regularization of neural networks using dropconnect. *ICML*,

[37] Krizhevsky, A. (2012). cuda-convnet. http://code.google.com/p/cuda-convnet/

[38] Theano: A Python framework for fast computation of mathematical expressions.

[39] Krizhevsky, A., Sutskever, I. and Hinton, E. G. (2012). Imagenet classication with deep convolutional neural networks. 1106–1114.

[40] Mackay, C. J. D. (1995). Probable networks and plausible predictions - a review of practical bayesian methods for supervised neural networks. *Bayesian methods for backpropagation networks, Springer.*,

[41] Weigend, S. A., Rumelhart, E. D. and Huberman, A. B. (1991). Generalization by weight elimination with application to forecasting. *NIPS*,

[42] Hinton, E. G., Srivastava, N., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *CoRR, abs/1207.0580*,

[43] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507.

[44] Van Der Maaten, L., Postma, E., and Van den Herik, J. (2009). Dimensionality reduction: a comparative.

[45] Salakhutdinov, R., Mnih, A., and Hinton, G. (2007). Restricted Boltzmann machines for collaborative filtering. *Proceedings of the 24th international conference on Machine learning, ACM*, 791–798.

[46] Larochelle, H. and Bengio, Y. (2008). Classification using discriminative restricted boltzmann machines. *Proceedings of the 25th international conference on Machine learning, ACM*, 536–543.

[47] Larochelle, H., Mandel, M., Pascanu, R., and Bengio, Y. (2012). Learning algorithms for the classification restricted boltzmann machine. *Journal of Machine Learning Research*, *13*, 643–669.

[48] Xie, P., Deng, Y., and Xing, E. (2015). Diversifying restricted boltzmann machine for document modeling. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM*, 1315–1324.

[49] Hinton, G. E. and Salakhutdinov, R. R. (2009). Replicated softmax: an undirected topic model. *Advances in Neural Information Processing Systems*, 1607–1614.

[50] Srivastava, N., Salakhutdinov, R. R., and Hinton, G. E. (2013). Modeling documents with deep boltzmann machines. *arXiv preprint arXiv:1309.6865*,

[51] Boureau, Y.-l., Cun, Y. L., et al. (2008). Sparse feature learning for deep belief networks. *Advances in Neural Information Processing Systems*, 1185–1192.

[52] Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1985). A learning algorithm for boltzmann machines. *Cognitive Science*, *9*(1), 147–169.

[53] Bengio, Y. (2009). Learning deep architectures for ai. *Foundations and trends in Machine Learning*, *2*(1), 1–127.

[54] Hubel, D. H. and Wiesel, T. N. (1963). Receptive fields of cells in striate cortex of very young, visually inexperienced kittens. *J. neurophysiol*, *26*(994), 1002.

[55] Weng, J., Ahuja, N., and Huang, T. S. (1992). Cresceptron: a self-organizing neural network which grows adaptively. *IEEE IJCNN, International Joint Conference on Neural Networks*, *1*, 576–581.

[56] Weng, J. J., Ahuja, N., and Huang, T. S. (1993). Learning recognition and segmentation of 3-d objects from 2-d images. *IEEE Proceedings, Fourth International Conference on Computer Vision*, 121–128.

[57] Riesenhuber, M. and Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature neuroscience*, *2*(11), 1019–1025.

[58] Lawrence S, Giles CL, Tsoi AC, Back AD (1997) Face recognition: A convolutional neural network approach. *IEEE transactions on neural networks*, *8*(1), 98–113.

[59] LeCun Y, Bengio Y. (1998) The handbook of brain theory and neural networks. *Convolutional Networks for Images, Speech, and Time Series*, 255–258.

[60] Lee H, Grosse R, Ranganath R, Ng AY (2009) Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. *Proceedings of the 26th annual international conference on machine learning. (ACM)*, 609–616.

[61] Boureau YL, Bach F, LeCun Y, Ponce J (2010) Learning mid-level features for recognition. *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. (IEEE)*, 2559-2566

[62] Huang FJ, Boureau YL, LeCun Y, , et al. (2007) Unsupervised learning of invariant feature hierarchies with applications to object recognition. *Computer Vision and Pattern Recognition, 2007. CVPR07. IEEE Conference on. (IEEE)*, 1–8

[63] Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning.*, 448-456.

[64] Nair V, Hinton GE (2010) Rectified linear units improve restricted boltzmann machines. *Proceedings of the 27th international conference on machine learning (ICML-10).*, 807-814.

[65] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. *California Univ San Diego La Jolla Inst for Cognitive Science.* (No. ICS-8506)

[66] Rubinstein, R. (1999). The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability.*, *1*(2), 127190.

[67] Graham, B. (2014). Fractional max-pooling. *arXiv preprint arXiv:1412.6071.*,

[68] Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806.*,

[69] Selvaraju RR, et al. (2016) Grad-cam: Visual explanations from deep networks via gradient-based localization. *arXiv preprint 1610.02391 v3.*,

[70] Zhou B, Khosla A, Lapedriza A, Oliva A, Torralba A (2016) Learning deep features for discriminative localization. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*, 2921-2929.

[71] Lin M, Chen Q, Yan S (2013) Network in network. *arXiv preprint arXiv:1312.4400.*,

[72] Bourlard, H. and Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics.*, *59*(4-5), 291-294.

[73] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research.*, *11*(Dec), 3371-3408.

[74] Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research.*, *15*(1), 1929–1958.

[75] Bock C, Poole G, Parker P, Gottwald T (2010) Plant disease severity estimated visually, by digital photography and image analysis, and by hyperspectral imaging. *Critical Reviews in Plant Sciences.*, *29*(2), 59107.

[76] Castelvecchi D (2016) Can we open the black box of ai? *Nature News*, *538*(7623), 20.

[77] Balu A, et al. (2016) Learning localized geometric features using 3d-cnn: An application to manufacturability analysis of drilled holes. *arXiv preprint*,

[78] Koenning SR, Wrather JA, , et al. (2010) Suppression of soybean yield potential in the continental united states by plant diseases from 2006 to 2009. *Plant Health Progress*, *10*

[79] Sladojevic S, Arsenovic M, Anderla A, Culibrk D, Stefanovic D (2016) Deep neural networks based recognition of plant diseases by leaf image classification. *Computational intelligence and neuroscience 2016.*

[80] Ubbens JR, Stavness I (2017) Deep plant phenomics: A deep learning platform for complex plant phenotyping tasks. *Frontiers in plant science 8.*

[81] Stoecklein D, Lore KG, Davies M, Sarkar S, Ganapathysubramanian B (2017) Deep learning for flow sculpting: Insights into efficient learning using scientific simulation data. *Scientific Reports 7*

[82] Dumoulin V, Visin F (2016) A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285.*

[83] Kingma D, Ba J (2014) Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

[84] Hartman GL, et al. (2015) Compendium of soybean diseases and pests. *American Phytopath Society)*

[85] Chiang KS, Bock CH, Lee IH, El Jarroudi M, Delfosse P (2016) Plant disease severity assessment how rater bias, assessment method, and experimental design affect hypothesis testing and resource use efficiency. *Phytopathology.*, *106*(12), 1451-1464..

[86] Singh A, Ganapathysubramanian B, Singh AK, Sarkar S (2016) Machine learning for high-throughput stress phenotyping in plants. *Trends in plant science.*, *21*(2), 110-124.

[87] LeCun Y., Bengio Y., Hinton G (2015) Deep learning. *Nature*, *521*(7553), 436-444

[88] Mohanty S. P., Hughes DP, Salath M (2016) Using deep learning for image-based plant disease detection. *Frontiers in plant science 7.*

[89] Berkooz, G., Holmes, P., and Lumley, J. L., (1993). The proper orthogonal decomposition in the analysis of turbulent flows. *Annual review of fluid mechanics*, *25*(1), 539-575.

[90] Schmid, P. J., (2010). Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, *656*, 5–28.

[91] Schmid, P. J., (2011). Application of the dynamic mode decomposition to experimental data. *Experiments in Fluids*, *50*(4), 1123-1130.

[92] Sarkar, S., Lore, G. K., Sarkar, S., Ramanan, V., Chakravarthy, R. S. and Phoha, S. (2015). Early detection of combustion instability from hi-speed imagesvia deep learning and symbolic time series analysis. *In Annual Conference of The Prognostics and Health Management*, 1–9

[93] Ghosal, S., Ramanan, V., Sarkar, S., Chakravarthy, S. and Sarkar, S. (2016). Detection and Analysis of Combustion Instability From Hi-Speed Flame Images Using Dynamic Mode Decomposition. *Proceedings of the ASME DSCC, 2016.*,

[94] Sarkar, S., Lore, G. K., Sarkar, S. (2015). Early Detection of Combustion Instability by Neural Symbolic Analysis on Hi-speed Video. *Symposium in Neural Information Processing Systems (NIPS)*, 1–9

[95] Akintayo, A., Lore, G. K., Sarkar, S. and Sarkar, S. (2016). Prognostics of Combustion Instabilities from Hi-speed Flame Video using A deep Convolutional Selective Autoencoder. *International Journal of Prognostics and Health Management*, *7*(23), 1–14.

[96] McManus, K., Poinsot, T., and Candel, S. (1993). A review of active control of combustion instabilities. *Progress in energy and combustion science*, *19*(1), 1–29.

[97] Smith, D. A., and Zukoski, E. E., (1985). Combustion instability sustained by unsteady vortex combustion. *American Institute of Aeronautics and Astronautics.*,

[98] Candel, S., (2002). Combustion dynamics and control: Progress and challenges. *Proceedings of the combustion institute*, *29*(1), 1-28.

[99] Sarkar, S., Lore, K. G., Sarkar, S., Ramanan, V., Chakravarthy, S. R., Phoha, S., and Ray, A., (2015). Early detection of combustion instability from hi-speed flame images via deep learning and symbolic time series analysis. *Proceedings of Annual Conference of the Prognostics and Health Management Society, (San Diego, CA).*,

[100] Kullback, S. and Leibler, R.A., (1951). On information and sufficiency. *The annals of mathematical statistics 22*(1), 79–86.